AFRL-IF-RS-TR-2002-304
Final Technical Report
November 2002

GETTING WHAT YOU WANT: ACCURATE DOCUMENT FILTERING IN A TERABYTE WORLD

University of Massachusetts Amherst

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** NOVEMBER 2002	**3. REPORT TYPE AND DATES COVERED** Final Sep 99 – Sep 02

4. TITLE AND SUBTITLE GETTING WHAT YOU WANT: ACCURATE DOCUMENT FILTERING IN A TERABYTE WORLD	**5. FUNDING NUMBERS** C - F30602-98-C-0110 PE - 62702F PR - 4594 TA - IH WU - 12
6. AUTHOR(S) Jamie Callan	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Prime: University of Massachusetts Amherst, Department of Computer Science, Amherst Massachusetts 01003-4610 Sub: CMU, Language Technical Institute, Pittsburgh PA 15213-8213	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFEA 32 Brooks Road Rome New York 13441-4114	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2002-304

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Daniel A. Ventimiglia/IFEA/(315) 330-3937/ Daniel.Ventimiglia@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

This document describes information retrieval research techniques to users who are interested in receiving filtered information for specific topics or domains. The research includes techniques for setting document filtering thresholds, for adaptive learning and for filtering based upon user feedback in order to refine the document filtering process.

14. SUBJECT TERMS Document Filtering, Relevance Filtering, Adaptive Filtering, Data Redundancy, Information Retrieval, Data Mining	**15. NUMBER OF PAGES** 79
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

List of Tables

1. Introduction

Information filtering systems monitor document streams to find documents that match information needs specified by user profiles. Information filtering is an “old” technology in the sense that it has been used in commercial and government software applications for more than thirty years. For most of that time the main requirement was to find *potentially relevant* documents in a stream of mostly irrelevant documents. This goal made sense when there were relatively few relevant documents, and when the matching technology was relatively primitive. However, those conditions changed dramatically in the 1990s. The stream of documents to be filtered is growing exponentially, and exact-match Boolean technology is being replaced by more accurate statistical matching techniques. Filtering systems are beginning to find more *potentially relevant* documents than are useful for many tasks.

The coming decade will be characterized by access to document streams of a terabyte or more per day. The documents a person has time to read will become a smaller fraction of the total document stream. If the current generation of filtering technology is applied to such a rapidly growing stream of documents, the “false hits” will consume an ever larger portion of the analyst’s time, and many of the “true hits” will merely repeat information found in previously seen documents. The occasional nugget of useful new information will become increasingly difficult to find.

Contract F30602-98-C-0110, “Getting What You Want: Accurate Document Filtering in a Terabyte World” (abbreviated here to the “*Accurate Document Filtering*” contract) was established to address this problem. The key ideas were a transition from judging documents by *relevance* to judging them by *utility*, an emphasis on machine learning algorithms that learn more accurate descriptions of information needs, and development of optimization techniques for probabilistic filtering systems. The focus was primarily on techniques that require minimal interaction from a user, for example, analysts interacting by specifying what is good and what is not good about examples found by the system. The goal was a system that interacts much as an intelligent human assistant would.

The research was to be organized around four system requirements. The system:

- Might rank by *relevance*, but must also interact with the analyst in terms of *utility (novelty)*;
- Must learn from its successes and mistakes;
- Must be very accurate, probably by using highly structured filtering queries; and
- Must be efficient enough to filter a terabyte a day for several thousand profiles on a small number of “commodity” (UNIX workstation, or Intel) processors.

This report describes the work done under the Accurate Document Filtering contract. The next chapter reviews the recent history and state-of-the-art in document filtering. Chapters 3 and 4 discuss the research conducted under the contract. Chapter 5 briefly describes the prototype system that was created. Chapter 6 summarizes the lessons learned during the contract research. Chapter 7 lists the publications created under the contract. Chapter 8 contains references.

2. Adaptive Information Filtering

The dominant method of filtering information for most of the last thirty five years was based on the Exact Match retrieval model and Boolean queries. Exact match systems can be implemented easily, they can be optimized to run extremely quickly, and it is easy to explain to a user why a particular document was or was not matched. Boolean queries allow people to express complex information needs very precisely and are popular with highly experienced users. These characteristics caused many filtering systems to remain "exact match" through the 1990s, when most ad-hoc information retrieval systems were shifting to statistical models.

It is well-known that information retrieval systems based on statistical models (vector-space (Buckley, et al. 1995), probabilistic (Robertson, et al., 1995), inference network (Callan, et al., 1995), language models (Zhai and Lafferty, 2001b)) are about twice as effective as systems based on exact match models. Statistical models allow "soft match" or "fuzzy match" of queries to documents, which makes them more forgiving of queries that don't do exactly what their authors intended. For example, the query "multimedia AND software AND NOT (Microsoft OR IBM OR Apple)" may be intended to express a desire for information about multimedia software from companies other than Microsoft, IBM or Apple, but it actually expresses a stricter constraint: Matched documents can't even *mention* Microsoft, IBM, or Apple. If an exact match model is used, few or no documents will be matched, because many of the desired documents are likely to also briefly mention competing products from Microsoft, IBM, or Apple.

Systems based on statistical models are also easier to use because people can express their information needs using free-text (phrases and sentence fragments) in addition to Boolean queries. For example, an information need might be expressed as "Iran Iraq war, 'chemical weapons'".

Another advantage of statistical models is that queries can be created automatically, based on automatic analysis of relevant and non-relevant documents. The Tipster program and the TREC Routing track, both started in the early 1990s, demonstrated quite clearly that filtering queries/profiles[1] created by automatic analysis of relevant and non-relevant documents were considerably more accurate than profiles created manually (Croft, et al., 1993). These early efforts were based on 'batch" analysis of large sets of documents, i.e., given a set of 300-1000 training documents the task was to create a profile. This approach to creating profiles modeled environments in which profiles changed rarely (because over a long period of time it would be possible to accumulate large amounts of training data for a profile). Although less dynamic than

[1] *Profile* and *query* are used somewhat interchangeably in the information filtering literature, but typically *profile* indicates a description of a long-term information need (e.g., in an information filtering system), and *query* indicates a description of a short-term information need (e.g., in an ad-hoc search engine). The different terminology indicates that a profile is based on more information than a typical ad-hoc query, for example, it might be influenced by user feedback given over a period of time. Operationally, both profiles and queries are expressed in a query language.

many users would like, it demonstrated the potential of statistical models for information filtering tasks.

Information filtering systems based on statistical models began to appear in the mid 1990s (e.g., Callan, 1996). Since then there has been a rapid transition to statistical models among researchers studying this area. The best forum for research on the topic has been the TREC Filtering tracks during TREC-8 (Hull and Robertson, 1999), TREC-9 (Robertson and Hull, 2001), and TREC-10 (Robertson and Soboroff, 2002).

When information filtering systems are based on statistical models, a new problem arises. Exact match models and Boolean queries either match a document or do not. Statistical retrieval models implement a "soft" or "fuzzy" match, so many documents match many queries to some degree. The match may be very strong or very weak, but often for a very large number of documents it is not zero. Filtering systems based on statistical retrieval models must therefore decide whether the match between document and query is strong enough for the document to be delivered to the user. This is usually described as a problem of setting dissemination thresholds. The problem is complicated for two reasons. First, there is no clear pattern in document scores for different profiles. A score of 0.4732 for one profile may be very good, but may be very bad for another profile. A second reason is that different people have different needs. One person may want to see every document related to a topic ("high Recall"), but another user may want to see only the most relevant documents ("high Precision"). Hence, dissemination thresholds must be set on a per-profile, per-user basis.

Information filtering systems that learn or tune profiles and thresholds over time, based upon periodic feedback from a person, are called *adaptive information filtering* systems. Adaptive information filtering systems were a fairly new research area when the contract began. Now they are the mainstream approach to information filtering by the research community, and they are beginning to be adopted commercially. The mainstream view of adaptive filtering systems has also shifted from relatively unresponsive systems that updated profiles rarely based on large amounts of training data to very responsive systems that update profiles incrementally whenever information from the user becomes available.

We include in our definition of adaptive filtering an additional task: learning to recognize which documents are novel and which are redundant relative to documents delivered previously for a particular profile. We call this task *novelty detection*. Novelty detection is not ordinarily considered a component of an adaptive filtering system. The current thinking is that novelty is an inherent property of a document; this is reflected in research on Topic Detection and Tracking (TDT) (Allan, 2002) and in the organization of the TREC-2002 Novelty track. The research done under this contract suggests that novelty is based on a person's interest and experience, in which case it cannot be learned without explicitly modeling the user. Given a user history, sufficient training data, and a relatively well-defined topic a system can learn one person's novelty criteria. This more constrained approach to novelty detection, which fits easily into an adaptive filtering system, offers a much better chance for success than the alternatives.

Developing experimental systems is a challenge when it is not possible to observe the intended users (in this case intelligence analysts), data stream, and operational environment. The contract research modeled the filtering environment with open source data provided to the research community by the National Institute for Standards and Technology (NIST) under its TREC program.

A corpus of documents was sorted chronologically, to simulate a document stream (e.g., a newswire). The filtering system was initialized with a set of information needs (often TREC topics, written by retired intelligence analysts), and sometimes with a small set of positive examples (relevant documents) for each information need. The system transformed these automatically into filtering profiles (queries). During filtering, documents would be considered one-at-a-time, in chronological order. The filtering system would compare the document to each profile and make decisions about whether to deliver the document to the users associated with each profile. When a particular document was disseminated for a particular profile, the system would receive feedback about whether its decision was correct. The system had an opportunity to revise the profile based on the (positive or negative) feedback prior to receiving the next document.[2] This process continued until each document in the stream had been considered. The system's performance over the entire stream of documents was measured using a variety of measures, including Precision, Recall, and various Utility metrics (Hull and Robertson, 1999; Robertson and Hull, 2001). This filtering environment is not a perfect model of an operational environment, but it is perhaps the best that can be done in the absence of real data and real users.

2.1 InRoute and YFilter

One of the first information filtering systems based on a statistical model was the InRoute system, which was based on a Bayesian inference network model of information filtering (Callan, 1996). The roots of the Bayesian inference network are information needs (the set of profiles known to the system), the internal nodes correspond to nested query operators, and the leaf is the document currently being filtered. Nodes are connected by directed arcs that represent dependencies. Beliefs propagate through the network, from the leaf to the roots, resulting in a measure of how strongly the system believes the current document is to satisfy each information need.

Although InRoute is based on a powerful model of statistical inference, every element of the model can be mapped to traditional data structures and relatively simple algorithms. A simple architecture diagram is shown in Figure 2.1.

The contract research was initially performed with the InRoute system. However, in 1999 the Principal Investigator switched universities, and this change revealed problems in the InRoute software license. A new information filtering system was implemented using off-contract funds. The new system, called YFilter, was based on a similar model and had equivalent capabilities but

[2] If a document was not disseminated, no feedback was given

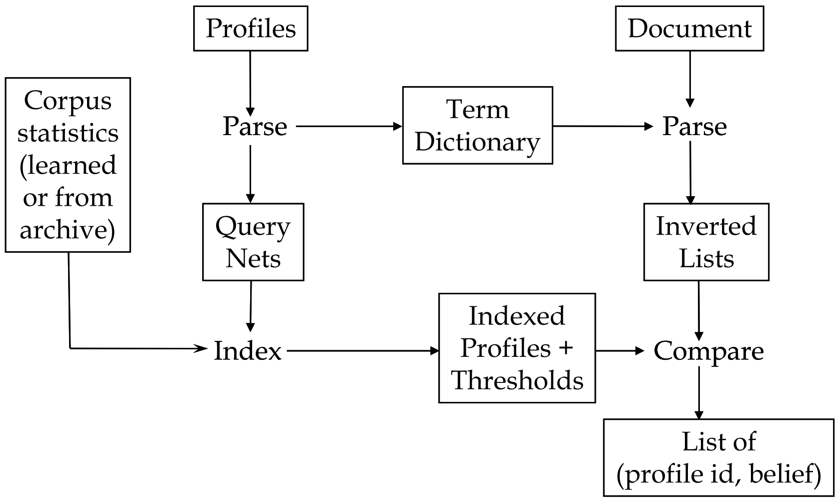

Profiles
Document
Corpus statistics (learned or from archive)
Parse
Term Dictionary
Parse
Query Nets
Inverted Lists
Index
Indexed Profiles + Thresholds
Compare
List of (profile id, belief)

Figure 2.1. **The InRoute information filtering system software architecture.**

no such licensing restrictions. YFilter was used through the remainder of the contract. YFilter runs under Unix (Sun Solaris) and Windows NT operating systems. Most of the research described in this report was done on an ordinary single-processor personal computer running Windows NT.

3. Learning Information Needs & Preferences

Most people find it difficult to create accurate queries for ad-hoc search. A typical pattern is for a person to tune a query to improve Precision, because “false hits” (non-relevant documents that were delivered) are easily observed. Often a consequence is that the query becomes increasingly narrow, so that many relevant documents are not delivered. Recall deteriorates, but the user does not know it, because the user does not see undelivered documents. Studies show that people tend to overestimate the effectiveness of their queries, especially Boolean queries, because they can observe Precision increasing but cannot observe Recall decreasing.

Anecdotal evidence suggests that intelligence analysts spend considerable time developing Boolean queries for information filtering. However, once a query is created and tuned, it is not necessarily adjusted or updated often after it is put into use. This pattern is not surprising – analysts are busy people with many responsibilities – but it means that the query is not responsive to changing language patterns.

Research on text classification and on document routing has shown that when a learning algorithm has sufficient training data, it can usually produce probabilistic classifiers and routing queries that are much more accurate than the best classifiers or queries created manually. The learning algorithms can identify useful patterns that an analyst might overlook, and it can be run automatically so that decisions about which patterns are predictive get updated on a regular basis.

A contract goal was to develop algorithms that could learn highly accurate queries, dissemination thresholds, and user preferences from a moderate amount of interaction with the user. This collection of information is called a *profile*, because it consists of more information than a typical *ad-hoc* query, and to emphasize that it represents a long-term information need. Improved information need representations improve the accuracy of the filtering system in identifying documents that a person would consider relevant to the task at hand.

The research on developing more accurate filtering profiles was broadly organized around three approaches. One approach was to identify efficient incremental profile-learning algorithms that provide the effectiveness of the “batch-oriented” query-creation algorithms used in TREC Routing experiments. A second approach was to improve the effectiveness of filtering queries by introducing more structure, for example more use of phrase, proximity, and Boolean operators. A third approach was to learn information about user preferences, for example dissemination thresholds that enable different users to receive different numbers of documents, depending upon their precision and recall needs. Each approach is discussed in more detail below.

3.1 Incremental Profile Learning

An incremental algorithm for learning filtering profiles has two tasks:

- **Term selection:** Select a set of terms to include in the profile; and
- **Term weighting:** Set (or adjust) the query term weights for each term in the profile.

These tasks are sometimes accomplished with the same algorithm, for example weighting all of the candidate terms and then selecting the best *N* terms.

An incremental profile learning algorithm can be invoked each time a positive or negative training document becomes available, or it can be invoked periodically, for example after each *N* new documents arrive. The research conducted under this contract was based on invoking the algorithm immediately as soon as each training document became available. This choice increases the responsiveness of the system, but it is only practical for very efficient algorithms, hence the research was confined to algorithms that could be implemented very efficiently.

Four learning algorithms were studied during the contract:
- Rocchio,
- Rocchio with modified idf,
- Mutual Information, and
- Language Models.

Each algorithm is described in more detail below.

3.1.1 Rocchio

The Rocchio algorithm is a version of the well-known Perceptron learning algorithm (Rocchio, 1971). It uses training data to determine weights for each feature in a feature vector (i.e., each term in a query). The basic Rocchio algorithm addresses the term-weighting task of profile learning. The basic Rocchio algorithm does not address the term selection task, however a common choice is to use the Rocchio weights for ranking candidate terms (e.g., the terms in the initial query, the relevant documents, and the non-relevant documents), and then to select the *N* terms that have the highest weights. In the research reported here, term selection was based on Rocchio weights, and query length (i.e., the number of terms selected) was based on the amount of training data available.

The Rocchio algorithm for computing term weights is usually described as a batch-oriented algorithm in which an initial query is adjusted using a set of labeled training examples, for example, documents marked relevant and not relevant. For an initial query Q, a set *R* of documents marked relevant and a set *NR* of documents marked non-relevant, the Rocchio formula is often described as shown below:

$$Q' = \alpha\,\mathrm{Q} + \beta\frac{1}{|R|}\sum_{d\in R} d - \gamma\frac{1}{|NR|}\sum_{d\in NR} d \qquad (3.1)$$

where *d* is a vector of tf.idf weights representing a document, and Q' is the query with learned term weights. Usually $\alpha + \beta + \gamma = 1$; the exact mixture is determined heuristically, often based on the amount of training data.

The Rocchio algorithm was first described in the mid 1960's, but fell out of favor because it was erratic when little training data was available, as was common in most ad-hoc relevance feedback environments. It came back into favor in the early 1990's for use in document routing applications, where large amounts of training data are available (Buckely, et. al, 1994), and in information filtering applications, where *eventually* the amount of training data becomes large. Rocchio remains a competitive algorithm today. There is a simple incremental implementation of Rocchio that can be applied on a per-document basis, that can be computed very efficiently, and that is guaranteed to produce exactly the same results as the batch-oriented version of the algorithm [Allan, 1996; Callan, 1998].

The incremental version of Rocchio was used as the baseline condition in the research reported here.

3.1.2 Rocchio and Modified IDF

One of the oldest observations in information retrieval is that the most "useful" terms are those that appear with medium frequency (Luhn, 1958): Common terms are not helpful for discriminating among documents, and very rare terms don't occur often enough to be generally useful. This hypothesis has been tested periodically, but has never yielded good results. We revisited it in this contract.

The inverse document frequency (*idf*) component of tf.idf term weights favors rare terms and penalizes common terms. A new term weight called *ydf* was created that favored terms in the middle of the frequency range. Ideally ydf is a convex function that reaches its maximum at the point of maximum utility in the range of term frequencies. The ydf weight was defined as shown below:

$$ydf_{t,d} = -idf_{t,d} \cdot \log(idf_{t,d}) - (1 - idf_{t,d}) \cdot \log(1 - idf_{t,d}) \tag{3.2}$$

$$idf_{t,d} = \kappa \frac{\log\left(\dfrac{N_d + 0.5}{df_{t,d}}\right)}{\log(N_d + 1)} \tag{3.3}$$

where N_d is the number of documents delivered before document d, $df_{t,d}$ is the number of documents containing term t, and κ is a constant controlling the point of maximal utility. Figure 3.1 shows the standard idf metric, and the effect of setting κ to 0.75 and 1.0.

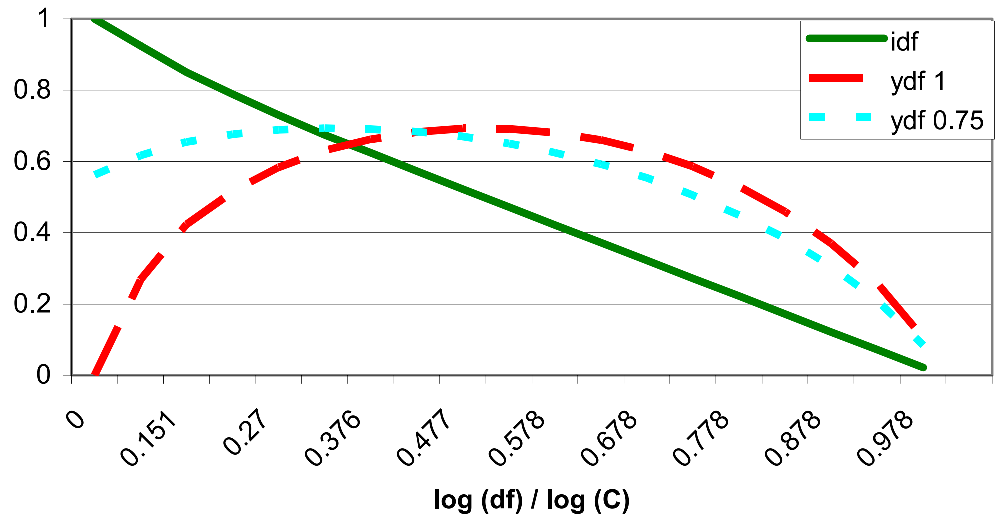


Figure 3.1. **Using ydf to favor words with middle range term frequencies: The relationship between document frequency, idf, and two forms of ydf.**

The new ydf weight is an alternative to the traditional idf weight. In the work reported here, it was combined with a standard Okapi-style BM25 weight to produce the final term weight.

$$T_{t,d} = \frac{tf_{t,d}}{tf_{t,d} + 0.5 \cdot 1.5 \cdot \frac{doclen_d}{avgdoclen}} \tag{3.4}$$

$$w_{t,d} = T_{t,d} \cdot ydf_{t,d} \tag{3.5}$$

The standard incremental Rocchio algorithm was tested with the modified idf weight in the adaptive filtering thread of the TREC-9 Filtering track (Robertson and Hull, 2001; Zhang and Callan, 2001c). The evaluation was performed using two different sets of profiles (OHSU and MeSH) and two different evaluation metrics (T9P, T9U).

The 63 OHSUMed profiles were similar in size and complexity to profiles used in prior TREC Filtering tracks. The 5,000 MesH profiles were based on MeSH (Medical Subject Headings) controlled vocabulary terms. The set of 5,000 MeSH profiles was two orders of magnitude larger than the number of profiles ordinarily studied by researchers; it was a real test of a system's ability to scale up to problems of realistic size. Only four research groups were able to submit results for the complete set of MeSH profiles in the TREC-9 evaluation; the remaining groups, some with computationally intensive algorithms, were given the option of working with a 10% sample. The set of 5,000 profiles was not a problem for the combination of the YFilter system and the incremental Rocchio profile-learning algorithm.

Average Utility / Precision					
T9P Metric		**T9P Metric (.M field)**		**T9U Metric**	
System	**Score**	**System**	**Score**	**System**	**Score**
Ok9f1po	0.294	**CMUDIR16**	0.279	KUNa2T9U	17.3
Ok9f2po	0.288	CMUCAT	0.224	KUNa1T9U	16.8
CMUDIR14	0.267	IOWAF003	0.138	ok9f3uo	10.7
FDUT9AF3	0.265			ok9f1uo	9.7
FDUT9AF1	0.264			FDUT9AF2	9.6
KUNa1T9P	0.258			**CMUDIR15**	9.3
FDUT9AF3	0.249			reliefs1	1.1
KUNa2T9P	0.231			IOWAF001	–5.9
CMUCAT3	0.213			kddaf903	–35.3
reliefs2	0.168			kddaf905	–35.5
antadapt002	0.102			kddaf906	–35.9
antadapt001	0.088			kddaf904	–36.4
				pircT9U1	–55.7
				pircT9U2	–69.1

Table 3.1: **TREC-9 Adaptive Filtering Track results for the set of 63 OHSU profiles. The CMUDIR entries are YFilter using the incremental Rocchio algorithm.**

The document set consisted of Medline documents published over a period of about 4 years and 9 months. The first 9 months of documents was available to a system as training data. Systems were tested on their ability to find relevant documents in the following 4 years of documents.

The T9P metric required a system to deliver *at least* an average of one document per month for each profile, whether or not there were actually any relevant documents during that period. This is a model of an environment in which analysts check periodically that profiles are performing as expected. It also has the added benefit that profiles acquire new training data periodically. The T9U metric awarded 2 points for delivering a relevant document and deducted 1 point for delivering a non-relevant document; a system could get a score of zero by simply not delivering any documents for any profiles.

The experimental results for the OHSU profiles are summarized in Table 3.1. The results for the YFilter system are displayed in bold, for easier identification. YFilter did very well when its performance was optimized for the T9P metric. Its performance was a little less competitive when optimized for the T9U metric, but the performance still compared well with all but one filtering system. One reason for the differing performance was that the T9U metric did not require a minimum delivery ratio. The Rocchio algorithm is more effective when it has more training data. The T9P metric forced the system to disseminate documents periodically, which had the effect of "buying" training data even when the system did not want to. In the long run, this strategy was more effective.

MeSH Profiles, Full		MeSH Profiles, Sample			
T9P Metric		T9P Metric		T9U Metric	
System	Score	System	Score	System	Score
ok9f2pm	0.419	**CMUDIR12**	**0.363**	ok9f1us	46.5
CMUDIR11	**0.359**	FDUT9AF6	0.356	ok9f3us	40.1
FDT9AF5	0.351			FDUT9AF7	29.3
CMUCAT2	0.303			**CMUDIR13**	**26.7**
				IOWAF002	12.9

Table 3.2: **TREC-9 Adaptive Filtering Track results for the MeSH profile set. The CMUDIR entries are YFilter using the incremental Rocchio algorithm.**

The results for the MeSH profiles are summarized in Table 3.2. There were actually two sets of MeSH profiles. The original, complete set contained nearly 5,000 profiles, each corresponding to a MeSH controlled vocabulary code. During the evaluation it became clear that some systems would be unable to handle the entire set of 5,000 profiles, so a subset of 500 profiles was created (the "sample" set). As with the OHSU profiles, YFilter did well when optimized for the T9P metric, and less well when its performance optimized for the T9U metric. These results must be viewed with caution, however, because so few systems participated in evaluations with the MeSH data.

Filtering systems were not evaluated specifically on efficiency, and indeed few systems reported their running times, except anecdotally. The running time for YFilter, including the time required to learn profiles adaptively and filter documents to them, was about 5 minutes for the OHSU profile set and about 6 hours with the complete set of MeSH profiles. Experiments were conducted on a mid-range Pentium III computer running Windows NT.

Subsequent analysis indicated that, although the overall quality of the results was good, the ydf term weight variation had very little effect (positive or negative). A standard idf weight yielded very similar results in our post-TREC experiments. As a result, subsequent experiments used the standard idf term weight.

3.1.3 Mutual Information

An alternative to discriminative learning (e.g., Rocchio) is to compare the probability of seeing a word in a relevant document with the probability of observing it in a random document. Terms that are highly correlated with relevant documents are candidate query terms; other terms are not. Like Rocchio, this approach uses training data to determine the strength of correlation between relevance and each feature in a feature vector.

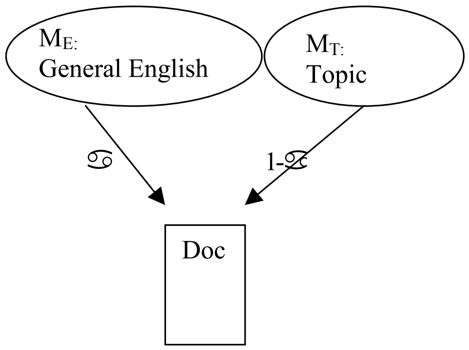


Figure 3.2. **A mixture model to generate on topic documents.**

The correlation between a term and relevance can be measured with the pointwise Mutual Information metric. Pointwise Mutual Information (Manning & Schutze, 1999) is defined as shown below:

$$I(x, y) = \log \frac{p(x, y)}{p(x)p(y)} \tag{3.6}$$

where x represents a candidate term, and y indicates whether the document is relevant. A mutual information value of 0 means that the presence of the term is completely independent of whether the document is relevant. Higher values mean that the presence of a term is highly correlated with the document being relevant.

The pointwise Mutual Information measure can be used in two ways:

1. To rank candidate terms for term selection, with another method (e.g., Rocchio) used to calculate the weights of the selected terms, or
2. To rank candidate terms for term selection and to calculate term weights for selected terms.

Both approaches were explored.

The Mutual Information (MI) algorithm was compared to the Rocchio algorithm in experiments with TREC data (e.g., 89,000 1989 AP Newswire documents, TREC Topics 51-100). The profiles produced by the incremental Rocchio algorithm were always at least as accurate as profiles created by the incremental MI algorithm. This result is consistent with what others have reported in ad-hoc retrieval environments, so this line of research was not pursued further.

3.1.4 Language Models

Probabilistic language models are used widely in speech recognition and have shown promise for ad-hoc information retrieval (Ponte and Croft, 1998; Lafferty and Zhai, 2001). The strong

(1) Calculate the General English language model:

For each word w_i

$$P(w_i \mid M_E) = \frac{tf\_i}{\sum_{j\text{:all words in corpus}} tf\_j}$$

(2) Calculate the Topic language model using the EM algorithm to maximize the likelihood of all relevant documents

a. Initialize Topic language model to a uniform model

$P(w_i \mid M_T) = 1/n$

where n is the number of unique words that appear in the relevant documents

b. EM step:

Iterate the following steps until changes in $P(w_i \mid M_T)$ are small enough for that iteration:

For each word in relevant documents:

$$T\_tf_i = rtf_i * \frac{(1-\alpha) * P(w_i \mid M_T)}{(1-\alpha) * P(w_i \mid M_T) + \alpha * P(w_i \mid M_E)}$$

$$P(w_i \mid M_T) = \frac{T\_tf_i}{\sum_{i\text{:all words in relevant documents}} T\_tf_i}$$

Figure 3.3. Training Algorithm for Language Model.

theoretical foundation of language models enables us to build a variety of new capabilities. Current research on using language models for information retrieval tasks is focused on developing techniques similar to those used in speech recognition. However the differing requirements of speech recognition and information retrieval suggest that major adaptation of traditional approaches to language modeling is necessary to develop algorithms that will be accurate in the real world.

The problem of learning profiles for adaptive filtering can be addressed with a mixture of generative language models, which is arguably a more accurate model of the information filtering task. As shown in Figure 3.2, each relevant document can be assumed to be generated by a combination of two language models: A general English model M_E, and a user-specific topic model M_T. For example, if the user is interested in "Star Wars", in a relevant document we expect words such as "is" and "the" will come from the general English model, and words such as "star" and "wars" will come from the topic model.

When doing user profile updating for query expansion and adjusting term weights, the filtering system should focus on learning M_T to find words that are very informative for deciding whether the document is on topic or not. Given a fixed value of α (usually a very high value, such as

0.95), we can train M_E to maximize the likelihood of all documents processed (relevant and non-relevant), and train M_T using the EM algorithm to maximize the likelihood of all the relevant documents processed. A sketch of the training algorithm is given in Figure 3.3.

This new mixture model will pick words where P(w| relevant) / P(w | general English) is very high. Because the task of profile updating is to provide a profile that can separate relevant documents from non-relevant documents, such words ought to be more discriminative, and thus good candidates for being added to user profiles. Similar techniques were developed for ad-hoc retrieval independently (Zhai and Lafferty, 2001).

This language modeling approach was evaluated in the TREC-10 Adaptive Filtering task. The document stream consisted of about 800,000 documents from a 1996-1997 Reuters newswire corpus. The usual types of information need profiles were not available for this corpus, so profiles were simulated with 84 Reuters categories. Robertson and Soberoff (2002) provide more details about the evaluation environment and metrics.

Four runs were submitted to the adaptive filtering task using Rocchio or the language modeling approach ("LM") for profile updating, and a Maximum Likelihood Estimation method for setting dissemination thresholds (Section 3.3.5). The results are reported below.

	Run 1	**Run2**	**Run3**	**Run4**
Profile learning method	Rocchio	Rocchio	LM	LM
Optimization metric	T10SU	T10F	T10SU	T10F
T10SU score	0.144	0.143	0.081	0.080
T10F score	0.273	0.275	0.158	0.163
Number of profiles with scores above mean score	55	41	32	21

The Rocchio method was more effective than the language modelling approach in these experiments, which was a surprise. Our hypothesis was that the language model would be more effective because of its stronger theoretical foundation.

Analysis of the results indicated several problems. One problem was that these experiments combined a language-modeling approach to learning profiles with the standard Okapi BM25 algorithm for determining how well each document matched a profile. In retrospect it is clear this was a mistake. The language-modeling approach is not biased against common words, so it will add words to a profile that the idf weight in the Okapi BM25 algorithm will subsequently discount. The Okapi BM25 algorithm is a much better match for the Rocchio algorithm.

A second problem is apparent from studying how Precision, Recall, and Utility metrics varied during filtering. Precision and Recall tended to improve during filtering, but Utility decreased.

This means that the profiles (terms and term weights) were improving during filtering, as more training data was seen, but the threshold was set too high and got worse over time. The threshold-updating algorithm was based on strong assumptions about the distribution of document scores for relevant and non-relevant documents. These assumptions (discussed in Section 3.3.5) have held with other datasets (Zhang and Callan, 2001), but not with this one. One possibility is that the Reuters categories were not a good method of simulating filtering profiles. This possibility is supported by the generally poor performance of a wide range of filtering systems on this dataset (Robertson and Soberoff, 2002).

We conclude that the language-modeling approach has promise, but that it should only be used with a language-modeling algorithm for determining how well documents match each profile. Unfortunately, most of the current language-modeling algorithms for determining how well queries match documents tend to be computationally expensive, thus not appropriate for an adaptive filtering environment. The combination of the Rocchio algorithm for learning profiles and the Okapi BM25 algorithm for determining how well documents match profiles is probably the best combination of effectiveness and efficiency.

3.1.5 Assessment

The learning algorithms were studied at different times during the contract, and under different conditions. There was no direct comparison of all four approaches on a common dataset. However, one of the algorithms, Rocchio, was used as a baseline condition in experiments with the other three algorithms. The use of a common baseline throughout the contract allows several conclusions to be drawn.

The Rocchio algorithm was consistently the most effective profile-learning algorithm tested. This result was surprising, but it is consistent with research done by other laboratories during this same time period. Rocchio is effective and computationally efficient.

The language-modeling algorithm appeared to be a good fit for the filtering task, but if it is to be effective, it must be paired with a comparable algorithm for determining how well a document matches a profile. The profiles learned by the language-modeling approach were clearly incompatible with the Okapi-style BM25 term weights. Research on a language-modeling approach to filtering fell outside the contract, and was likely to produce a slower system, so it was not pursued. However, the underlying idea, i.e. that a document can be modeled by a mixture of language models, is powerful. This idea recurs in our work on novelty detection (Section 4).

3.2 Structured Queries

Most relevance feedback algorithms used in IR learn a linear discriminator. These algorithms essentially adjust weights on a set of features in a linear classifier. The features in an information filtering task tend to be query terms, resulting in very simple, if often powerful, statistical classifiers. This is sometimes called the "bag of words" or "unstructured" model of information needs. However, these learning algorithms are also compatible with learning more structured queries, should that be required, because queries tend only to involve weights at the top-level of the query. A complex structured query looks to the learning algorithm like a linear classifier with complex features.

There is evidence that more powerful features would result in more accurate queries. Profiles created by expert users, such as librarians and intelligence analysts, tend to be highly structured. That is, they contain a wide range of query operators, they contain multiple ways of expressing the same concept, they employ grouping operators such as *OR* and *SYNONYM* to indicate variant expressions of a concept, and they use *AND* and proximity operators to indicate constraints among the concepts in a query. Although there has been substantial research on learning queries automatically, based upon user feedback about which documents were and were not relevant (*"relevance feedback"*), there has been much less work on learning highly structured queries similar to what people create.

The contract research explored three approaches to introducing more structure into the profiles of an adaptive filtering system: use of phrases, use of named-entities, and use of hierarchically organized profiles. All three have delivered significant improvements in accuracy in Routing or batch filtering environments. The goal of the research was to develop equivalent methods that would work incrementally, as relevant documents are encountered in the filtering stream.

3.2.1 Phrases

Substantial research indicates that phrases and proximity operations are strongly correlated with filtering effectiveness. For example, the phrase "white house" is far more specific than either the Boolean query 'white AND house' or the free text query 'white, house'. Very effective techniques have been developed for identifying important phrases and collocations in relevant documents, but they are all computationally complex. They are appropriate for batch-oriented environments, but not for interactive filtering environments where a person wants the system to change its filtering behavior quickly based upon new information.

An incremental approach to incorporating phrases into profiles was developed as part of the contract research. Whenever a document was identified as relevant, all two-word sequences ("bi-grams") that didn't contain stopwords were considered to be candidate phrases. For example, for the text fragment "…hit a home run yesterday…" the phrases "home run" and "run yesterday" would be generated. The bi-grams "hit a" and "a home" would not be generated because "a" is a stopword. Scores were assigned to phrases using the usual profile-learning algorithm (in this case, Rocchio).

Tests with TREC Filtering track corpora showed no improvement with the addition of phrases. This result was a surprise, because prior research in the closely-related TREC Routing track showed that phrases could yield significant improvements in accuracy (Allan, et al., 1996). Post-mortem analysis suggested that the problem was the incremental nature of the adaptive filtering task. The successful use of phrases during the TREC Routing task was due in part on the ability to select phrases from a relatively large set of relevant documents. During adaptive filtering no such set is available. Instead, phrases are created from a very small set of relevant documents, and then additional relevant documents are acquired over time. The total number of relevant documents available for training the profile (and for learning phrases) never approaches the amount of data available during TREC Routing tasks. The statistical (bi-gram) approach to generating phrases is simply too "noisy" for the small amount of training data available.

3.2.2 Named Entities

Another approach to adding structure to profiles is to enable the use of named-entity features. Named-entities are a more abstract class of features that represent classes of concepts such as PERSON, ORGANIZATION, COMPANY, LOCATION, DATE, and MONEY (monetary amount). Named-entity features can be recognized by a named-entity recognizer, such as BBN's IdentiFinder (Bikel, et al., 1997; Bikel, et al., 1999), during document parsing. Query operators can be added to the user profiles to enable matching of named-entitities in documents. The combination of named-entity indexing and named-entity query operators enables profiles such as "hijack NEAR/20 *PERSON", which matches any occurrence of a person name that occurs within 20 words of the word "hijack".

The utility of named entity tagging for adaptive information filtering was investigated under the contract. YFilter was modified to recognize named-entity (ENAMEX) tags in document text, and to include new query language operators (*PERSON, *LOCATION, etc) that can match them. These adjustments enable the use of named-entity tags in filtering profiles that are created manually or automatically, for example, "Oklahoma City bombing *PERSON" to find documents that mention people names in conjunction with the Oklahoma City bombing. BBN provided a copy of its IdentiFinder software for detecting named entities in documents (Biken, et a., 1997; Bikel, et al., 1999).

A series of experiments was conducted with TREC Filtering Track data. Profiles were created initially without named-entity features, based upon TREC topics created by NIST, but the automatic profile-learning software was allowed to add such features if it appeared that they were correlated with relevance for a particular profile.

In tests with a set of 50 profiles, the profile learning software added named-entity operators to only 10 (out of 50) profiles. Presumably named-entities were not correlated highly with relevance in the other profiles. In the 10 profiles for which they were added, named-entity features caused essentially no change in performance metrics for 9 of the 10 profiles, as measured by a wide range of metrics (e.g., Precision, Recall, F1, T9P, T9U). One profile had a relatively larger improvement, in part because three named-entity operators (*DATE, *PERCENT, *LOCATION) were added to it.

This result was initially disappointing, but upon further investigation it is not surprising. *Many* documents contain at least one named-entity of a particular type, for example, a person, so named-entity features tend to have low inverse document frequency (idf) weights. They are easy to match, and their effect on a document score is low.

The University of Massachusetts, which investigated the use of named-entity operators during its Tipster research in the early and mid 1990s, found that named-entity operators were most effective when used as arguments to proximity operators. For example, "*PERSON" might not be helpful, but "*PERSON NEAR/20 bombing" (i.e., find a person name within 20 words of the word "bombing") were effective.

Although creating such proximity operators automatically can greatly improve accuracy, doing so is very computationally expensive, because a very large number of term co-occurrences must be considered during profile learning. Usually every pair of terms that are fewer than N words apart would be considered by the profile-learning algorithm, for example, every pair of terms that occurs within 20 words of each other. Hundreds of term pairs, and hence hundreds of co-occurrence features, can be generated in an average-length document.

The prior experience learning phrases for use in profiles suggested that TREC-style adaptive filtering environments would not contain sufficient training data to enable the profile-learning program to identify useful co-occurrence relationships.

3.2.3 Hierarchical Profiles

In some situations information needs are organized hierarchically. For example, one profile might cover "European politics" and another one might cover "French politics". Any document that matches "French politics" could also serve as training data for "European politics". Hierarchical profiles are a frequent request in commercial environments.

In 2001 the TREC Filtering Track profiles were defined by Reuters newswire category codes, which are organized hierarchically. This data provided an opportunity to investigate whether the filtering system would be more accurate if it explicitly considered the hierarchical organization of the filtering profiles. For example, if a document did not match the "European region" profile, then it should not be considered for the "French exports" profile. Likewise, training data for the "French exports" profile could also be used to train the "European region" profile, increasing the amount of training data for, and the accuracy of, profiles that represent relatively broad, high-level topics. The research hypothesis was that this feature would improve accuracy in the TREC Filtering task.

YFilter was modified to support hierarchically organized profiles. Both the basic filtering system and the profile-learning system were modified. When filtering hierarchically the filtering engine won't consider a document for a child profile if the parent profile doesn't match. The profile-learning system can use training data for child profiles to also train parent profiles, thereby leveraging training data to improve accuracy.

The hierarchical approach to filtering was tested in the TREC 2001 Filtering Track. The filtering profiles corresponded to Reuters categories, which are organized hierarchically. We assumed that if a document belonged to a child category, it should also belong to the parent category. We used the following rules to take advantage of the hierarchical relationship between profiles when making the decision whether to deliver a document. If C_i is a child category of C_j, then:

- When a document d_t comes, first consider whether it should be delivered to C_j, and if so, *then* consider whether it should be delivered to C_i.

- If the document d_t was delivered to C_j and the system received negative feedback, do not deliver d_t to C_i

- If d_t was not delivered to C_j previously, but was delivered to C_i and the system received a positive feedback, deliver d_t to C_j

This procedure provides more training data for some profiles. For example, profiles 17, 34, and 45 started with 8 instead of 2 relevant training documents, which improved early stage accuracy.

One problem with this hierarchical approach is that the Reuters category assignments (i.e., the training data) are not consistent. Some documents are judged as relevant to a child category but non-relevant to the parent category. For example, documents 135,639, 24,269, and 26,015 belong to R18 (DOMESTIC MARKETS) but do not belong to R17 (MARKETS/MARKETING). It is well-known that human category assignments are not perfectly consistent, and any algorithm that uses them must compensate for noise in the training data. Using hierarchical structure of the categories helps to solve this problem to some extent during training, but similar mistakes in testing data will penalize the filtering system.

Analysis of the TREC 2001 results suggest that hierarchical learning and filtering capabilities improved TREC Filtering track results for a few profiles. For example, the system had 8 documents for learning the Marketing profile, instead of just 2, because it was able to incorporate documents from profiles beneath Marketing in the classification hierarchy. However the number of profiles that was affected was small, and so the overall effect on the filtering system was not particularly noticeable. Participants in the track generally agreed that the Reuters categories were a poor substitute for real information needs, and so it is difficult to draw firm conclusions from this test about the effectiveness of hierarchically organized profiles in adaptive filtering.

3.2.4 Assessment

Three techniques were explored to improve the expressive power of profiles learned incrementally during adaptive filtering: use of phrases, use of named-entities, and use of hierarchically organized profiles. All three were expected to work well, and have delivered significant improvements in accuracy in Routing or batch filtering environments. However, none of them were particularly effective in the adaptive filtering experiments conducted under this contract.

One problem with the research conducted under the contract was that it was restricted primarily to TREC Filtering data and the TREC filtering methodology, which may not be represent the conditions in government environments. In particular, the TREC Filtering tracks have recently focused on use of very small amounts of training data and a relatively sparse stream of relevant documents, i.e., the needle in the haystack type of problem. The lack of sufficient training data is clearly an obstacle to assessing the effectiveness of complex features, such as phrases and co-occurrence relationships. Complex features are more precise than single terms, but they are also much more rare, so there are fewer opportunities to learn appropriate term weights.

Likewise, the Reuters 2001 corpus is probably not a good model for the use of hierarchical filtering profiles in government environments. The use of hierarchical profiles clearly improved the amount of training data (the number of relevant and non-relevant documents) for some Reuters categories. However, the Reuters 2001 categories are not a good model of information needs in an adaptive filtering environment, so it isn't possible to draw firm conclusions about the effectiveness of hierarchical profiles.

The clearest conclusion from these tests is that these methods should be tested in government environments to assess their effectiveness with "real world" data.

3.3 Dissemination Thresholds

Document filtering systems that rely on statistical or probabilistic models must decide how well a document must match a profile before it is shown to the user. This is expressed as the problem of setting dissemination thresholds, i.e. the minimum score or criterion that a document must attain to be disseminated. Until recently there was relatively little work on how to learn these from incremental relevance feedback at the same time that the profile (which determines a document's score) is also being learned.

Most statistical retrieval models are based on tf.idf formulas that favor query terms that i) are frequent in the document, and ii) are rare in the corpus. One characteristic of tf.idf algorithms is that it is difficult to say in advance what a "good" document score would be for a particular query. A "good" score depends on the length of the query, the particular query operators (if any) used in the query, and the relative frequency of the terms in the corpus. The concept of "goodness" is also user-dependent. Users don't all want the same amount of information. Some want to see everything ("high recall"), while others are more concerned about not having their time wasted ("high precision"). An appropriate dissemination threshold depends upon the query structure, word frequency patterns in the corpus, and user preferences.

One way for people to express their dissemination preferences to the filtering system is as a desired threshold as a probability of relevance, for example "Show me every document that has more than a 50% chance of being relevant." This constraint could be indicated graphically as a knob or slider bar that a person adjusts to get a desired balance of Precision and Recall. The system's job is to translate this user requirement into a dissemination threshold appropriate for the current query and corpus.

The contract research studied four types of algorithm for setting dissemination thresholds:

- **Up-and-Down:** Increase threshold when a non-relevant document is delivered, decrease it gradually otherwise;
- **Binning:** Bin documents based on their scores, learn probability-of-relevance for each bin, set threshold based on desired probability of relevance;
- **Regression:** Use logistic regression to learn a mapping from document scores to probability-of-relevance, set threshold based on desired probability of relevance; and
- **Score-modeling:** Model document scores as a mix of exponential and normal distributions, determine probability-of-relevance for each distribution, set threshold based on desired probability of relevance.

In addition, research was done on the effect of using unjudged documents as training data. Each of these research topics is described in detail in the following sections.

3.3.1 Up-and-Down

The Up-and-Down threshold algorithm is a very basic approach to setting dissemination thresholds. It was developed primarily as a very simple baseline algorithm. It assumes that a small amount of training data is available, and that a minimum delivery requirement is expected (e.g., the profile must deliver at least one document per month). This algorithm was tested in TREC-9 (Zhang and Callan, 2000).

Given the initial relevant documents, a profile is created using the Rocchio algorithm (Rocchio, 1971). The new profile is then used to score the other documents in the training set, without any feedback about whether the documents are actually relevant. The initial threshold is set to allow the top $\varphi(M+\delta)$ documents in the training dataset to be disseminated, where φ is the minimum delivery ratio the system must meet, M is the number of documents in the training set, and δ is an adjustment that lowers the initial threshold slightly so that the system will disseminate a little more aggressively in the beginning, thus obtaining more feedback for learning. δ was arbitrarily set to 2 in our experiments.

During operational use, each negative feedback increases the threshold. Otherwise, the threshold is always decreasing according to φ and the current profile's performance. The magnitude of an increase to a threshold is limited by $\mathrm{maxstep}$, which is empirically set to 0.005, so that a non-

```
if (doc$_i$ is delivered to profile$_k$) and (feedback is available)
   {
   if (doc$_i$ is not relevant)
      threshold =
         threshold –
         (max (maxstep, score (doc$_i$ , profile$_k$) - threshold) /
           sqrt (1 + number of feedback documents seen for profile$_k$));
   }
   else
      if (current delivery ratio $\varphi'$ < minimum delivery ratio $\varphi$)
         {
         if  (performance(profile$_k$) > 0)
            decrease_step = (Threshold - 0.4)* $\varphi$
         else
            decrease_step = (Threshold - 0.4) * ($\varphi - \varphi'$);

         threshold=max (threshold - decrease_step, 0.400);
         };
```

Figure 3.4: **An outline of the Up-and-Down threshold updating algorithm.**

relevant document with an extremely high score will not push the threshold too high, thus controlling the effects of outliers. Profile performance was measured with the TREC-8 F2 utility metric (Hull and Robertson, 1999). If a profile's F2 utility is positive, it is regarded as a good profile, and, therefore its threshold is decreased comparatively faster (Figure 3.4). The intuition is to disseminate more documents for good profiles, while keeping the delivery ratio for bad profiles just high enough to meet the minimum delivery requirement.

The Up-and-Down algorithm worked reasonably well in the TREC-9 Filtering task evaluation (Zhang and Callan, 2001c; Robertson and Hull, 2001). The main disadvantage of the algorithm is its use of so many constants, which must be set empirically. For the TREC-9 evaluations the constants were set using training data from TREC-6 and TREC-8 Filtering track evaluations, which worked reasonably well. However, in general this algorithm, although simple to understand and tune, is sufficiently ad-hoc that it would be difficult to trust in a very dynamic environment.

3.3.2 Binning

A research goal was to transform document scores into probabilities of relevance. For example, instead of returning a document score of 0.463523, which might be an excellent score for one

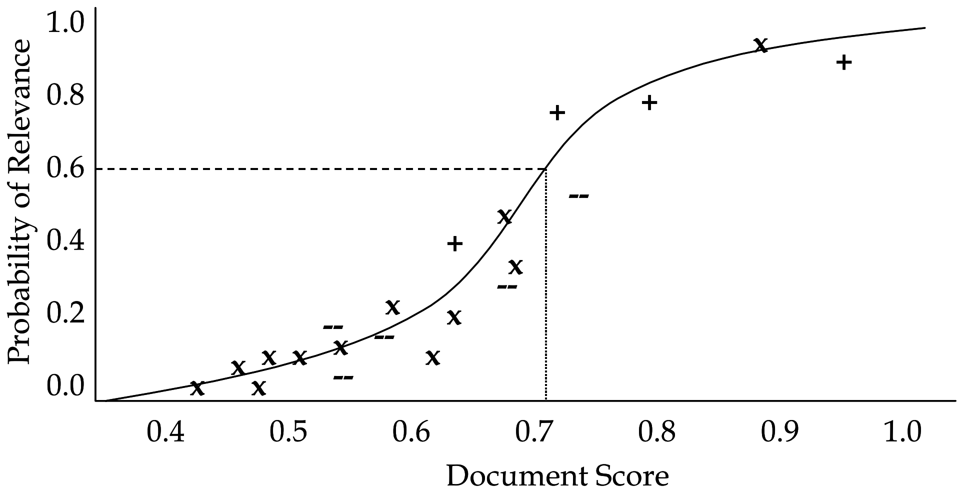


Figure 3.5. **The relationship between document scores and probability of relevance for a single profile. The curve indicates the underlying relationship, to be learned. An *x* indicates an unjudged document. A + indicates a document that a person judged relevant, and a – indicates a document that a person judged not relevant. The dotted lines show how to locate a dissemination threshold that delivers documents with a probability of relevance greater than or equal to 60%.**

profile and a poor score for another, one would prefer the system to indicate that there is a 65% probability that the document is relevant for a particular profile. This goal could be accomplished by changing the underlying retrieval model to produce probabilities, which is difficult; or by learning a profile-specific function that maps document scores to probabilities of relevance (e.g., Figure 3.5). The latter approach is the focus of this section and the following two sections.

Perhaps the simplest approach to mapping scores to probabilities of relevance is a binning algorithm. The range of potential document scores is divided into a series of bins. For example, one bin may cover the range 0.4 – 0.41, the next bin may cover the range 0.41 – 0.42, and so on. The bin size can be based on a score range (e.g., documents with scores in the range 0.40 – 0.41) or as a number of documents (e.g., each bin contains between 5 and 10 documents).

The binning algorithm assumes that there is an initial dissemination threshold. Its value can be set arbitrarily; if it is too high, it will be reduced later; if it is too low, it will be increased later.

During filtering, arriving documents are placed into the appropriate bin based on their scores. If a document is disseminated, we assume that an analyst will read it (eventually) and indicate whether the document was relevant or not relevant. The probability of relevance for each *bin* can be learned from this feedback.

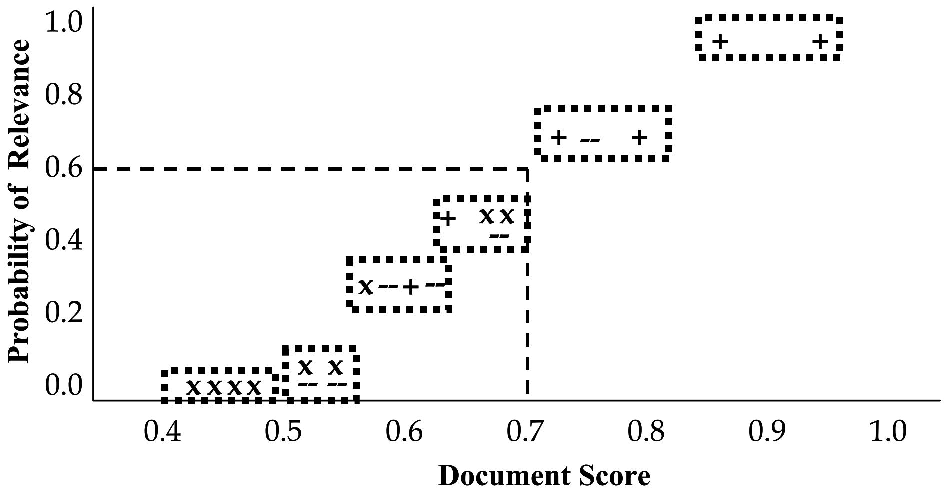


Figure 3.6. **An illustration of a simple binning algorithm. Each bin holds about 4 documents. An x indicates an unjudged document, a + indicates a document judged relevant, and a – indicates a document judged non-relevant. Probability of relevance is calculated for each bin. For example, the probabilities of relevance are 0%, 0%, 33%, 50%, 67%, and 100% (from left to right; assumes unjudged documents are ignored). The dotted lines indicate how to set a dissemination threshold based on a desired probability of relevance.**

$$P(\text{relevant} \mid \text{bin}_i) = \frac{\text{Relevant}_i}{\text{Judged}_i}$$

The probability of relevance for documents in the i'th bin is the number of relevant documents it contains divided by the number of judged documents it contains. Thus each bin has a probability of relevance.

If bins are set initially to cover different score intervals, e.g., 0.41—0.42, then initially most bins will have unknown probabilities of relevance, because they do not yet contain training data. An alternative is to start with a single bin that covers the entire score range, and then to divide a bin in half whenever it contains some number of documents, e.g., 10. Hybrid algorithms are also possible.

Figure 3.6 illustrates the binning algorithm. In this simple example the probability of relevance increases monotonically with the document score; in practice, probability of relevance will fluctuate up and down due to how scores are grouped. Such fluctuation can be handled heuristically, for example by choosing the lowest or highest bin that provides a desired probability of relevance, or by smoothing, for example using regression (covered in the next section).

The binning method was tested in experiments on the TREC-7 Filtering track data (Hull, 1999). The corpus was about 240,000 1988-1990 AP newswire documents. The profiles were TREC topics 1-50. The measure of effectiveness used at that time in the TREC Filtering community was the number of queries that produced an "above median" F1 measure (F1 is a metric that combines precision and recall), as compared with the systems that participated in the TREC-7 Filtering track. This metric measures how well a technique compares to its peers, irrespective of whether a particular query is "easy" or "hard". Typically the best systems are consistently above median.

The binning algorithm produced above median performance on 35-40 out of 50 queries. These results are encouraging because the binning algorithm has considerably lower computational complexity than most of the other algorithms.

A variant of the basic binning algorithm was tested in the TREC-8 Filtering task (Allan, et al., 2000). This variation disabled profiles that produced poor performance, and it used overlapping bins. Other forms of profile learning, e.g., adjusting terms and term weights, were disabled. The corpus was the 1992-1994 Financial Times. TREC topics 351-400 served as filtering profiles. The results were average. Overall, Recall was higher than for most of the other systems tested; Precision was a little lower. About two thirds of the profiles produced above median results; the other third produced very weak performance, dragging down the overall score for the system. These results are strong support for the binning approach to learning dissemination thresholds because learning was enabled *only* for the thresholds. Most of the other systems in the top group also dynamically adjusted the profile terms and term weights while filtering, which our system did not do.

Binning algorithms are extremely simple to implement, computationally efficient, and relatively effective. The only assumption about the transformation of document scores into probabilities of relevance is that it is a relatively monotonic relationship; otherwise it is a relatively non-parametric method, which makes it very general. The main weakness is that the lack of underlying assumptions about the form of the transformation (i.e., the shape of the curve) causes it to require more training data than some other methods. For example a dozen relevant documents might be necessary before a good dissemination threshold could be found for a desired probability of relevance. If a person later adjusts the desired probability of relevance, for example to make the profile deliver fewer documents, or more documents, the learning process must start again in a new document score neighborhood. The previous learning may not transfer to the new neighborhood.

3.3.3 Regression

Binning algorithms have several undesirable characteristics: They require a certain minimum amount of positive training data (relevant documents), they are somewhat difficult to use when the training data is noisy, and they are inefficient when users often change their delivery preferences. These disadvantages are caused by the “local” nature of binning algorithms. Binning algorithms confine their learning to a specific region, in this case, to a specific range of document scores.

An alternative is to learn the entire function that maps document scores to probabilities of relevance. Such a function would “smooth” the information learned by a binning algorithm, reducing the effects of noisy data, and coping more easily with users who change their delivery requirements often. In this section we present a regression-based algorithm that attempts to do just that.

The initial approach was based on sorting judged documents into bins, determining the probability of relevance within each bin, and then fitting a curve to a graph of probability of relevance vs. average score in the bin using logistic regression (Figure 3.5). It was quite successful for some percentage of the profiles, but consistency was an issue.

Probability of relevance is correlated with document scores for some profiles, but not others. A study of TREC-8 Filtering profiles was conducted to determine the extent to which it is possible to set thresholds that deliver a desired level of precision. For example, achieving 30% Precision is possible for nearly all profiles, given good thresholds. Some profiles can achieve 60% Precision with a good threshold, but for other profiles there is no threshold that delivers such precision. This problem affects all adaptive filtering systems. For example, in the TREC-7 Filtering track, the system from Claritech monitored its behavior and simply shut off profiles that could not achieve a given level of accuracy (Zhai, et al., 1999). For these profiles, no dissemination threshold will deliver the desired level of accuracy. Other modes of learning are required, to improve the profile terms, weights, and/or query structure.

From a computational perspective, Logistic Regression is impractical to run frequently due to its complexity. Our solution in operational environments was to update thresholds occasionally, after small batches (e.g., 1,000) documents were filtered. This solution produced reasonably accurate results at an acceptable computational cost.

The regression method was tested in experiments on the TREC-7 Filtering track data. The corpus was about 240,000 1988-1990 AP newswire documents. The profiles were TREC topics 1-50. The measure of effectiveness used at that time in the TREC Filtering community was the number of queries that produced an “above median” F1 measure (F1 is a metric that combines precision and recall), as compared with the systems that participated in the TREC-7 Filtering track. This metric measures how well a technique compares to its peers, irrespective of whether a particular query is “easy” or “hard”. Typically the best systems are consistently above median.

When the thresholds were learned by logistic regression, InRoute produced above median performance on 30-35 out of 50 queries. (The results vary within a range because measurements were taken after observing varying amounts of training data.) Although these results indicate that the regression approach can be effective, they are not particularly strong. The “binning” algorithm delivered better results on the same dataset, and it has considerably lower computational complexity than logistic regression.

3.3.4 Effect of "Near Miss" Documents

Typically adaptive information filtering systems use only disseminated documents as training data. Undisseminated documents may be used to learn corpus statistics such as average document length or inverse document frequencies (idf), but they usually are not used for profile-specific learning. However, unjudged document could be used for learning, for example by treating them as not relevant because they did not match the profile well enough to score above threshold. The research described in this section studied the effects of varying the use of judged and unjudged documents when learning dissemination thresholds. This research was conducted for the Binning and Logistic Regression threshold-learning algorithms only.

It is often assumed that training on roughly equal proportions of relevant and nonrelevant documents delivers the most accurate results. However, nonrelevant documents are more common than relevant documents, and unjudged documents (documents not disseminated for a particular profile) are more common still. If more nonrelevant documents, or unjudged documents, could be used in learning, dissemination thresholds might be learned more quickly or with less user intervention.

One research result was that it is not necessary to keep the numbers of judged relevant and nonrelevant documents balanced when learning thresholds. This result was not surprising for the binning algorithm, because adding more judged nonrelevant documents just adds more bins with low probability of being relevant; these bins are effectively ignored by the threshold-learning algorithm.

The more interesting result was that even for the logistic regression algorithm, adding more judged nonrelevant training data improved results. Ordinarily logistic regression is most effective when training data is spread evenly across the entire curve. Our hypothesis was that the judged nonrelevant scores tended to cluster in the vicinity where the threshold lies, hence providing a good fit in the region of interest, even if the mapping of scores to probabilities is distorted at the ends of the curve.

The effects of unbalanced training sets on the quality of learned thresholds is illustrated by the table below. The Precision of the average query consistently increased as additional judged non-relevant documents were added to the training data.

Training Data Relevant : Nonrelevant	**Set[3] Precision**	**Average[4] Precision**
1 : 1	0.1568	0.2519
1 : 2	0.2385	0.3142
1 : 4	0.3256	0.3998
1 : 9	0.4460	0.5057

Set precision and average precision both increased significantly when the amount of training data was increased *only* by adding additional judged non-relevant documents (and treating them as not relevant).

A second research result was that adding large numbers of "near miss" documents to the training set improves overall consistency and results. "Near miss" documents have scores just below the dissemination threshold, so they are not shown to the user. If "near miss" documents are treated as nonrelevant training data, accuracy increases. This result held for the Binning and Logistic Regression training algorithms. The effect on Logistic Regression was larger, bringing the accuracy of its thresholds up to the level of the Binning algorithm. This result is confined to training on documents with scores just below the dissemination threshold. There is little value in training on unjudged documents that are far from the threshold.

The table below summarizes the result of a representative experiment in which the goal was to learn profiles with an average precision of 40%. The bottom line represents the baseline experiment, in which none of the unjudged documents was used for training. The baseline Average Precision was 38.4% and, coincidentally, the Average Recall was similar, at 37.8%. The top line represents an experiment in which all of the unjudged documents were used for training, as was discussed above. The Average Precision improved to 42.8% and Recall dropped to 30.1%.

Threshold Setting Method:	Logistic Regression
Precision target:	0.4
Training set size:	60 judged (number of unjudged varies)
Ratio of Relevant to Retrieved judged documents:	1:5
Training corpora:	ap88, ap89
Test corpora:	ap90

[3] Set precision is a metric used at TREC. It puts all of the documents disseminated by a set of profiles into a pool, and then measures the percentage of relevant documents in the pool. Set precision can be skewed up or down dramatically based on the performance of a single good or bad profile.

[4] Average precision is determined by calculating the precision of each profile individually, and then averaging the precision values. Average precision is less affected by one particularly good or bad profile.

% Unjudged Documents Used for Training	Unjudged / Judged	% Non-relevant that are Unjudged	Set Precision	Set Recall	Average Precision	Average Recall
100.0%	65.4	98.5%	0.3699	0.2523	0.4275	0.3075
50.0%	32.7	97.0%	0.3715	0.2431	0.4264	0.2995
25.0%	16.3	94.2%	0.3835	0.2423	0.4685	0.3180
20.0%	13.1	92.9%	0.3780	0.2343	0.4606	0.3097
10.0%	6.5	86.7%	0.3789	0.2262	0.4658	0.3001
5.0%	3.3	76.5%	0.3714	0.2048	0.4636	0.2798
1.0%	0.6	39.2%	0.3721	0.1928	0.4657	0.2703
0.5%	0.3	24.0%	0.3696	0.2048	0.4540	0.2810
0.1%	0.1	5.08%	0.1397	0.2562	0.4166	0.3369
0.0%	0.0	0.0%	0.1319	0.2852	0.3840	0.3783

As the table is scanned from top to bottom, the percentage of unjudged documents that are used for training is reduced; only the highest scoring unjudged documents are used for training. As lower-scoring unjudged documents are ignored (for training), Average Precision rises, to 46%, while Average Recall stays relatively constant. When only 5% of the unjudged documents are used for training, Average Recall begins to fall, and when only 0.1% of the unjudged documents are used, Average Precision begins to fall.

The target Average Precision in these experiments was 0.4, modeling a user who wanted to see every document that had at least a 40% chance of being relevant. In this experiment, training on just a few of the best unjudged documents produced results close to what the user requested. Using the best "near miss" documents only increases the training costs by a small amount.

One might conclude from the Average Precision metric that there is little value to training on unjudged documents, because training on none of them delivers Average Precision values very close to what the user requested. However, metrics based on averages can mask problems in individual profiles. The Set Precision and Recall metrics are more sensitive to small numbers of profiles that have skewed Precision/Recall values, and the Set Precision/Recall values are considerably lower. Mixing a small number of unjudged documents into the training data produces a significant increase in the consistency of the Precision/Recall results from each profile, which is reflected in smaller differences between Set Precision/Recall and Average Precision/Recall.

One likely reason for this result is the nature of the learning task. The task is to learn a mapping of document scores, as determined by a particular profile, to probability of relevance. This task can be accomplished with any documents. The assumption that all unjudged documents are nonrelevant will be wrong occasionally, but rarely enough that the learning algorithms are not affected. The benefit of additional training data far outweighs the effects of the occasional error that is introduced into the training data.

This approach was not extended to learning the profiles themselves, which means that the filtering system used different training data for learning profiles and thresholds. Profile learning requires high-quality training data. Lower-quality training data appears to be sufficient for learning thresholds.

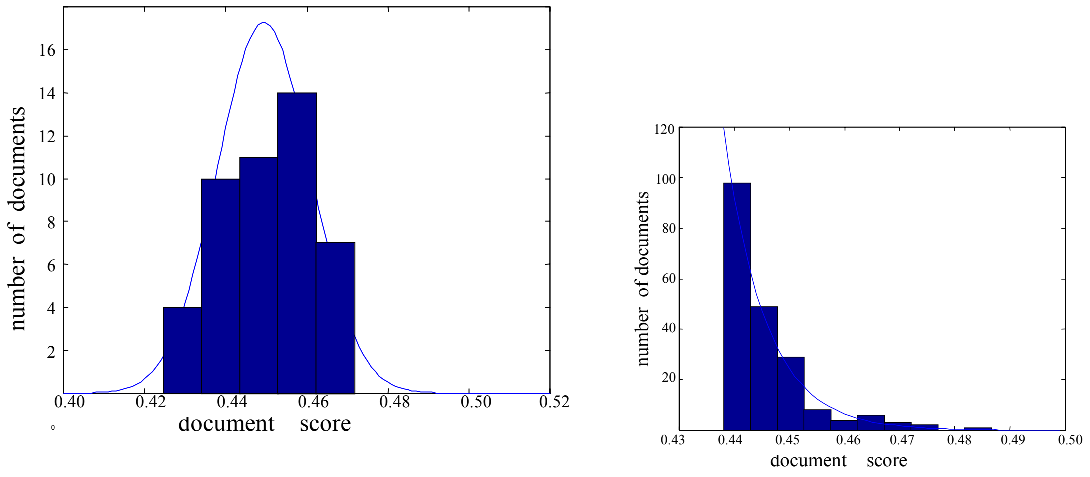
3.3.5 Score-Modeling

Prior research showed that the distribution of document scores from a statistical retrieval model could be modeled as a mixture of two distinct distributions, one for relevant documents and another for non-relevant documents (Manmatha, et al., 2001; Arampatzis and van Hameren, 2001). The basic idea is simple: Scores of relevant documents tend to fit a Gaussian (Normal) distribution, while scores of non-relevant documents tend to fit an Exponential distribution. Given a set of ranked retrieval results (e.g., from a search engine), it is possible to model the set of document scores as a mixture of scores drawn from the two distributions. The result is a mapping of document scores to probabilities.

This approach can be applied to information filtering, but the problem is somewhat different. Document scores are observed in an uncontrolled order, which makes the problem more challenging. However, explicit relevance information for some documents is available from the user, which may make it possible to learn the underlying distributions from less data. Arampatzis adapted the basic approach to an information filtering environment and got good results (Arampatzis, et al., 2001). However, this adaptation implicitly assumed that the training data accurately represents the distribution of relevant and non-relevant document scores. This assumption is not true in an adaptive filtering environment, because relevance information is obtained only for documents that are actually disseminated. This assumption produces thresholds that disseminate fewer documents than is optimal (i.e., Precision is maintained, but Recall is too low).

A new algorithm was developed for setting dissemination thresholds that, like the Arampatzis score modeling method, assumes that relevant document scores are distributed normally and non-relevant document scores are distributed exponentially (Arampatzis, et al., 2001). However, the new algorithm explicitly models the sampling bias, and uses the maximum likelihood principle to find unbiased estimates of the parameters. It jointly estimates the parameters of the two density distributions and the ratio of the relevant documents in the corpus. Experiments indicate that this new method produces dissemination thresholds that are significantly more accurate than the baseline score modeling method in some cases.

We call this new method Maximum Likelihood Estimation (MLE) for modeling score distributions. Section 3.3.5.2 describes our experimental methodology and our new algorithm for setting dissemination thresholds based on these score distribution. Section 3.3.5.3 reports experimental results. Finally, Section 3.3.5.4 summarizes the research on the use of Maximum Likelihood Estimation for modeling score distributions.



(a) **(b)**

Figure 3.7. **Density of relevant (a) and non-relevant (b) document scores: Topic5.**

3.3.5.1 Modeling Score Distributions

A Mathematical Model of Document Score Distributions

Usually only a small proportion of documents is relevant. Given an accurate model of the distribution of document scores for the top ranking documents (relevant and non-relevant), the dissemination threshold can be set accurately.

Several researchers suggest modeling score distributions using a Gaussian distribution for the scores of relevant documents and an exponential distribution for the top ranking non-relevant documents (Arampatzis, et al., 2001; Manmatha, et al., 2001). These distributions are modeled as:

$$P(score \mid R = r) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(score-u)^2}{2\sigma^2}}$$

$$P(score \mid R = nr) = \lambda e^{-\lambda(x-c)}$$

where:

u : mean of the Gaussian distribution
σ: variance of Gaussian distribution;
$1/\lambda$: variance of exponential distribution; and
c : minimum score a non-relevant document can get.

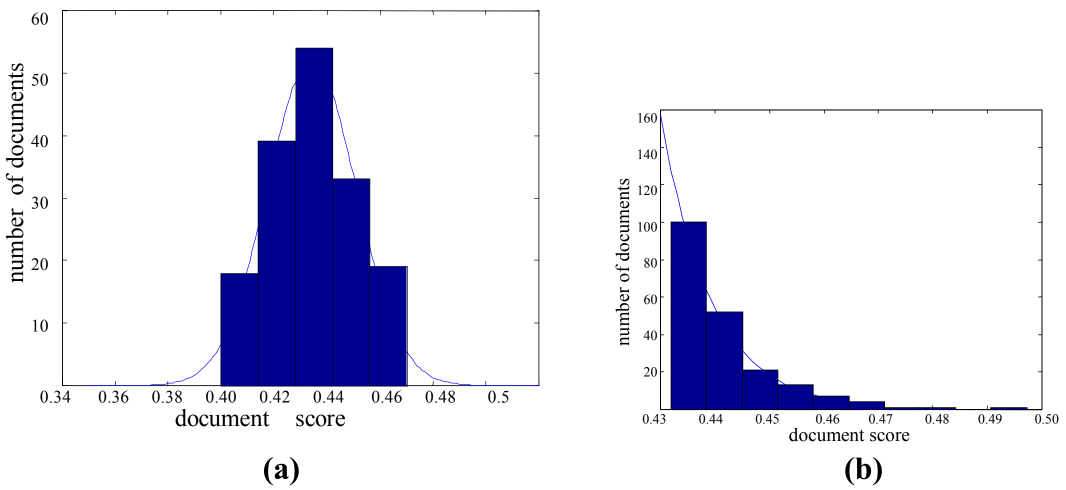


Figure 3.8. **Density of relevant (a) and non-relevant (b) document scores: Topic3.**

Analysis of experimental results from the TREC9 Filtering Track data (OHSU topics, OHSUMED collection) support this approach. Figures 3.7a and 3.8a illustrate how the normal distribution fits relevant document scores for two TREC9 topics. Figures 3.7b and 3.8b illustrate how the exponential distribution fits the top-100 non-relevant document scores for the same topics.

This model is not perfect, especially for low scoring documents. However, it can provide a relatively accurate estimate of the distribution of scores for higher-scoring documents, and we believe that in the information-filtering task the area of interest (i.e., the area in which thresholds are likely to be located) is among the higher-scoring documents.

Problem with the Basic Parameter Estimation Method

Given a set of scores, the basic parameter estimation method for normal and exponential distributions is a simple calculation of the mean and variance over training data (Arampatzis, et al., 2001). This approach to parameter estimation produces biased estimates, because in the information filtering task relevance information is available only for documents with scores above the dissemination threshold. When training data is restricted to scores above the threshold, the mean of the sample scores is likely to be higher than the real mean.

For example, for the profile in Figure 3.8a, the mean and variance of the Gaussian distribution (based on all of the relevant documents) are (0.4343, 0.0169). However, if training data is restricted to documents with scores above a fixed dissemination threshold of θ = 0.4435, the

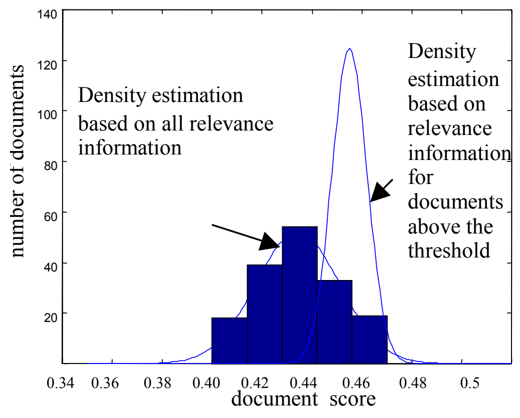


Figure 3.9. **Estimation of parameters for relevant document scores: Topic 3.**

mean and variance calculated using the basic score modeling method are (0.4551, 0007) (Figure 3.9).

In order to get an unbiased estimate of the distribution parameters, we must take into consideration the sampling constraint, i.e., the dissemination threshold. In an adaptive information filtering, the threshold is changing over time, so the problem becomes more interesting.

Maximum Likelihood Estimation of Score Distribution Parameters

Maximum Likelihood Estimation, an unbiased parameter estimation method, can be used to solve this problem. At a certain point in the filtering process, the filtering system has already delivered N documents to a user and relevance judgments of delivered documents are provided by the user. We can treat these documents as training data. For the i th delivered document with user feedback, let's represent it with a triple $(R_i, Score_i, \theta_i)$, where:

$$\begin{aligned} R_i &= \begin{cases} r & \text{for relevant document} \\ nr & \text{for non-relevant document} \end{cases} \\ Score_i &: \text{The score of document } D_i\text{; and} \\ \theta_i &: \text{The threshold when } D_i \text{ was delivered.} \end{aligned}$$

In order to describe the density distribution of the scores, 4 parameters are necessary: (u, σ, λ, p). The meanings of u, σ, and λ were described above. p is the expected ratio of relevant documents in the corpus *according to the model*. p does not represent the ratio in the corpus as a whole, because the exponential model fits only the top non-relevant scores. The model is focused on, and is most accurate modeling, the scores of the top-ranking documents, where thresholds are typically set. Given the observed training data, which are represented by a set of triples

$D = \{(R_i, Score_i, \theta_i), i = 1\, to\, N\}$, according to Bayes theorem, the most probable value of H = (u, σ, λ, p) is:

$$\mathrm{H}^* = \arg\max_{\mathrm{H}} P(\mathrm{H} \mid D) = \arg\max_{h} \frac{P(D \mid \mathrm{H}) P(\mathrm{H})}{P(D)} \tag{3.7}$$

For simplicity, we first assume that there is no prior knowledge of the distribution of *H* and treat the prior probability of *P(H)* as uniform. (We will revisit and remove the assumption in Section 3.3.5.2.) Because *P(D)* is a constant independent of *H*, it can be dropped. Thus the most probable *H* is the one that maximizes the likelihood of the training data.

$$\begin{aligned}(u^*, \sigma^*, \lambda^*, p^*) &= \arg\max_{(u,\sigma,\lambda,p)} P(D \mid (u, \sigma, \lambda, p)) \\ &= \arg\max_{(u,\sigma,\lambda,p)} \prod_{i=1}^{N} P(D_i \mid \mathrm{H}) \\ &= \arg\max_{(u,\sigma,\lambda,p)} \sum_{i=1}^{N} \log(P(D_i \mid \mathrm{H})) \\ &= \arg\max_{(u,\sigma,\lambda,p)} \sum_{i=1}^{N} \log(P(Score = Score_i, R_i \mid \mathrm{H}, Score > \theta_i))\end{aligned} \tag{3.8}$$

The second step of Equation 3.8 is due to the assumption that each document is independent; the third step is due to the fact that maximizing a function is equivalent to maximizing its logarithm; and the last step indicates the sampling constraints for training data. Notice that although the training data are not sampled randomly from the whole corpus, each individual training document is sampled randomly according to the conditional probability $P(Score= Score_i, R_i| H, Score>\theta_i)$. So parameters estimated based on Equation 3.8 are unbiased.

For each item inside the sum operation of Equation 3.8, we have:

$$\begin{aligned}P(Score = Score_i, R_i \mid \mathrm{H}, Score > \theta_i) &= \frac{P(Score = Score_i, Score > \theta_i, R_i \mid \mathrm{H})}{P(Score > \theta_i \mid \mathrm{H})} \\ &= \frac{P(Score = Score_i, Score > \theta_i \mid R_i, \mathrm{H}) P(R_i \mid \mathrm{H})}{P(Score > \theta_i \mid \mathrm{H})} \\ &= \frac{P(Score = Score_i \mid R_i, \mathrm{H}) P(R_i \mid \mathrm{H})}{P(Score > \theta_i \mid \mathrm{H})}\end{aligned} \tag{3.9}$$

The first step in Equation 3.9 is based on the definition of conditional probability; the second step is due to the chain rule of probability; and the last step is due to the fact that all training documents must have a score higher than the threshold.

For convenience, let $f_1(u, \sigma, \theta_i)$ be the probability of a document getting a score above threshold θ_i if it is relevant and $f_2(\lambda, \theta_i)$ be the probability of it getting a score above threshold if it is non-

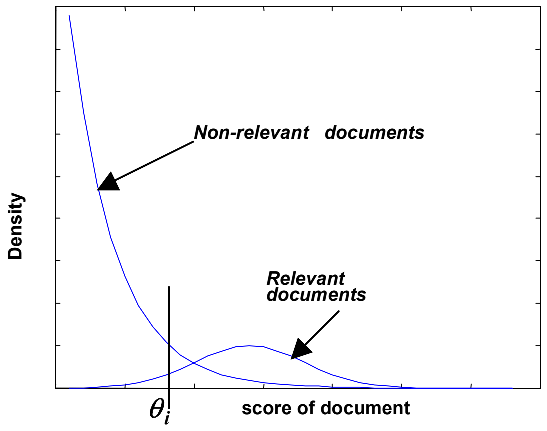


Figure 3.10. Distribution of scores of documents for a profile.

relevant. The probability of getting a score above θ_i can be calculated by integrating the density function from θ_i to positive infinity. That is:

$$\begin{aligned} f_1(u,\sigma,\theta_i) = P(Score_i > \theta_i \mid R_i = r) &= \int_{\theta_i}^{+\infty} P(Score = x \mid R_i = r)dx \\ &= \int_{\theta_i}^{+\infty} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-u)^2}{2\sigma^2}} dx \end{aligned} \tag{3.10}$$

$$\begin{aligned} f_2(\lambda,\theta_i) = P(Score_i > \theta_i \mid R_i = nr) &= \int_{\theta_i}^{+\infty} P(Score = x \mid R_i = nr)dx \\ &= \int_{\theta_i}^{+\infty} \lambda e^{-\lambda*(x-c)}dx = e^{-\lambda(\theta_i - c)} \end{aligned} \tag{3.11}$$

If we use $g(u,\sigma,\lambda,p,\theta_i)$ to represent the probability of a document getting a score above threshold θ_i, we get:

$$\begin{aligned} g(u,\sigma,\lambda,p,\theta_i) &= P(Score > \theta_i \mid \mathrm{H}) \\ & P(R = r \mid \mathrm{H}) \cdot P(Score > \theta_i \mid R = r, \mathrm{H}) \\ &= + P(R = nr \mid \mathrm{H}) \cdot P(Score > \theta_i \mid R = nr, \mathrm{H}) \\ &= p \cdot f_1(u,\sigma,\theta_i) + (1-p) \cdot f_2(\lambda,\theta_i)) \end{aligned} \tag{3.12}$$

An intuitive explanation of Equation 3.12 is illustrated in Figure 3.10. If we assume the sum of areas under the exponential and Gaussian curves is 1, then $p \cdot f_1(u,\sigma,\theta_i)$ corresponds to the area to

```
x = initial guess for the minimum
q = negative of gradient at x (search direction)
do {
        x = the minimal point along direction h
        q = a linear combination of new gradient and old q
} until convergence
```

Figure 3.11. **Sketch of the CG algorithm.**

the right of θ_i and below the normal distribution curve. $(1-p)\cdot f_2(\lambda,\theta_i)$ corresponds to the area to the right of θ_i and below the exponential distribution curve.

From Equations 3.8-3.12, we get:

$$(u^*,\sigma^*,\lambda^*,p^*) = \underset{(u,\sigma,\lambda,p)}{arg\,max} \sum_{i=1}^{N} LP_i \qquad (3.13)$$

where for relevant documents:

$$LP_i = -\frac{(Score_i - u)^2}{2\sigma^2} + \ln(p/(\sigma \cdot g(u,\sigma,\lambda,p,\theta_i)))$$

and for non-relevant documents:

$$LP_i = -\lambda(Score_i - c) + \ln((1-p)\lambda / g(u,\sigma,\lambda,p,\theta_i))$$

If the training data are random samples of the whole corpus (not true for filtering), θ_i =-∞ and thus $g(u,\sigma,\lambda,p,\theta_i)$ = 1. If we replace $g(u,\sigma,\lambda,p,\theta_i)$ in Equation 3.13 with 1, the optimal parameters will be the same as those proposed by (Arampatzis, et al., 2001). In other words, the basic score modeling method is a maximum likelihood estimation of the parameters if the training data are a random sample of the dataset. However, $g(u,\sigma,\lambda,p,\theta_i)$ is not equal to 1 because the sample is biased during filtering. We must use Equation 3.13 to find the unbiased parameters.

Parameter Optimization Using Conjugate Gradient Descent

There is no closed form solution for Equation 3.13, so numerical methods must be used. One such method is conjugate gradient descent (abbreviated *CG*). CG is an iterative minimization procedure for a function $f(x)$. A sketch of the CG algorithm is shown in Figure 3.11. At each step, x is the approximate solution, q is the search direction, and x is improved by searching for a better solution along direction q. More detailed descriptions are available in (Press, et al., 1992; MacKay, 2001).

The derivative of the right side of Equation 3.13 is used for calculating the search direction q. Taylor expansion is used to calculate the integration in Equation 3.10, and the Taylor expansion is guaranteed to converge.

3.3.5.2 Experimental Methodology

The baseline and unbiased ML methods of setting dissemination thresholds were evaluated in a series of experiments with TREC data. The experimental methodology for the TREC Filtering Track changes from year to year. In our experiments we used the TREC-9 experimental methodology (Robertson and Hull, 2001) in experiments with TREC-8 and TREC-9 Filtering data.

The advantage of this approach is that experimental results obtained with different datasets can be compared, because they were obtained with the same experimental methodology. However, a consequence is that our results with TREC-8 data are not directly comparable to results from systems participating in the TREC-8 Filtering Track. The evaluation measures, datasets, and experimental methodology are described in more detail below.

Evaluation Measure

One commonly used evaluation measure for a filtering system is linear utility, which corresponds to assigning a positive value or negative cost to each element in the category (Robertson and Hull, 2001).

	Relevant	**Non-relevant**
Delivered	R^{+} / A	N^{+} / B
Not Delivered	R^{-} / C	N^{-} / D

$$\text{Utility} = \mathrm{A} \cdot \mathrm{R}^{+} + \mathrm{B} \cdot \mathrm{N}^{+} + \mathrm{C} \cdot \mathrm{R}^{-} + \mathrm{D} \cdot \mathrm{N}^{-} \tag{3.14}$$

The variables R^{+}, N^{+}, R^{-}, and N^{-} are the number of documents in the corresponding category, and *A, B, C* and *D* are the benefit or cost associated with that category.
Filtering according to a linear utility function (Equation 3.14) is equivalent to setting the threshold at θ^{*}, where:

$$\frac{C-A}{B-D} \cdot \frac{p}{1-p} \cdot P(score = \theta^{*} \mid R_i = r) = P(score = \theta^{*} \mid R_i = nr)$$

The solution was solved by Arampatzis, et al. (2001) by:

$$\theta^* = \begin{cases} (b - \sqrt{\Delta})/a & \text{if } \Delta \geq 0 \\ +\infty & \text{if } \Delta < 0 \end{cases} \quad (3.15)$$

where:

$$\Delta = b^2 - ad$$

$$a = \frac{1}{\sigma^2}$$

$$b = \frac{u}{\sigma^2} + \lambda$$

$$d = \frac{u^2}{\sigma^2} - 2\log(\lambda \cdot p \cdot \frac{1}{\lambda\sqrt{2\pi}\sigma}) + 2\lambda \cdot c$$

If we let (A, B, C D) = (2, 1, 0, 0), the utility becomes[5]:

$$T9U' = 2 \cdot R^+ - N^+ \quad (3.16)$$

which is a slightly simplified variant of the T9U utility metric used in the TREC-9 Filtering track. This is the measure we used in our experiments. The corresponding delivery rule is:

$$\text{deliver if } P(R = r \mid score) > 0.33 \quad (3.17)$$

Data

Two different text corpora were used to test our algorithm: the OHSUMED dataset used in the TREC-9 Filtering track and the FT dataset used in the TREC-8 Filtering track.

The OHSUMED data is a collection from the US National Library of Medicine's bibliographic database (Hersh, et al., 1994). It was previously used by the TREC9 Filtering Track (Robertson and Hull, 2001). It consists of 348,566 articles from the subset of 270 journals covering the years 1987 to 1991. 63 OHSUMED queries and 500 MeSH headings were used to simulate user profiles. The relevance judgments for the OHSUMED queries were made by some medical librarians and physicians on the basis of output of interactive search (Hersh, et al., 1994). For the MeSH headings, assignment of a heading to a document by the National Library of Medicine is treated as equivalent to a positive relevance judgment.

In the TREC9 Filtering Track, it is assumed that the user profile descriptions arrive at the beginning of 1988, so the 54,709 articles from 1987 can be used to learn word occurrence (e.g.,

[5] The TREC-9 T9U measure is *Max* $(2R^+ - N+, MinU)$, where MinU=-100 for OHSU topics or –400 for Mesh topics. T9U sets a lower bound on profile utility, limiting the impact of poor profiles on overall filtering results. We have no need for the lower bound in our experiments, because all of the profiles get scores above MinU, so we opted for the simpler definition of utility

idf) statistics and corpus statistics (e.g., average document length). For each user, the system begins with a natural language description of the information need and 2 examples of relevant documents. The average numbers of relevant articles in the testing data are 51 for the OHSUMED topics and 249 for the Mesh headings.

The FT data is a collection of 210,158 articles from the 1991 to 1994 Financial Times. It was previously used by the TREC8 Filtering Track (Hull and Robertson, 1999). TREC topics 351-400 were used to simulate user profiles. The relevance judgments were made by NIST on the basis of pooled output from several searches.

For each profile, the system begins with two identified relevant documents and a natural language description of the information need, which is the title and description field of the corresponding TREC topic. The average number of relevant articles in the testing data is 36.

Methodology

The YFilter information filtering system (Zhang and Callan, 2001c) was used for our experiments. It processes documents by first removing unusual symbols (such as punctuation and special characters), excluding the 418 highly frequent terms listed in the default INQUERY stop words list (Broglio, et al., 1995), and then stemming using the Porter (1980) stemmer. Processed documents are compared to each profile using the BM25 tf.idf formula (Robertson, et al., 1995) to measure the similarity of the document and user profile.

For each topic, the filtering system created initial filtering profiles using terms from the TREC topic Title and Description fields, and set the initial threshold to allow the highest-scoring $\delta = 3$ documents in the training dataset to pass. For simplicity, the filtering profile term weights are not updated while filtering.

Because the first two relevant documents given to the system were not sampled under the constraint that their scores must exceed the dissemination threshold, their probabilities are simply $P(d_i \mid R_i = r)$, and the corresponding element of Equation 3.13 was changed to:

$$LP_i = -\frac{(Score_i - u)^2}{2\sigma^2} - \ln(\sigma)$$

In Section 3.3.5.1, for simplicity we set the prior probability of $P(\mathrm{H})$ to be uniform. However, in the real filtering task, especially during the early stage of filtering where there are only a small number of samples, this may cause problems. For example, if only non-relevant documents are delivered, the estimate of p will be 0 without a prior. Or, if all the relevant documents have the same score, the variance will be 0. Smoothing using a prior of parameters can solve these problems. The prior needn't be very accurate, because as the amount of sample data increases, the influence of the prior decreases. In our experiments, we set the prior of p as a beta distribution[6]: $p^{\varepsilon_1} \cdot (1-p)^{\varepsilon_2}$, which is equal to adding ε_1 relevant documents and ε_2 non-relevant documents sampled randomly for smoothing. The prior of σ is set to be $\exp(-\nu^2/(2\sigma^2))$, which is equal to adding ν^2 to the sum of the square of the variance of relevant documents [7]. We set $\varepsilon_1 = 0.001, \varepsilon_2 = 0.001, \nu = 0.005$ in our experiment, which is equivalent to adding 0.001 relevant documents and 0.001 non-relevant documents, and adding 0.005 to the sum of the square of the variance of Guassian distribution.

For each query set, 4 runs were carried out. The first run (the *baseline run*) used the parameter estimation method described in (Arampatzis, 2001). The third run used our maximum likelihood estimation. Both runs stopped delivering documents when Δ is negative (Equation 3.15). This is especially common at the beginning of filtering, so a minimum delivery ratio was introduced to avoid this problem. If a profile has not achieved the minimum delivery ratio, its threshold will be decreased automatically. This corresponds to the second and fourth runs. In our experiments, the minimum delivery ratio was set to let approximately 10 documents be disseminated to each profile.

3.3.5.3 Experimental Results

All of the experiments discussed in this report attempt to optimize the T9U′ Utility metric, which is consistent with recent TREC Filtering Track evaluation methodologies. Optimizing Utility can lead to higher Precision or Recall as a side-effect, but is not guaranteed to do so. In some cases the best way to increase Utility is to increase Precision (higher threshold); in other cases the best way to increase Utility is to increase Recall (lower threshold). We report all three metrics to provide greater insight into the experimental results, but we consider only Utility, the metric being optimized, when evaluating the results.

The experimental results for the OHSUMED dataset indicate that neither algorithm works well without using a minimum delivery ratio (Table 3.3, columns 1 and 3). Especially at the early stage of filtering, with only about 2 documents delivered, it is very hard to estimate the score distribution correctly. When $\Delta < 0$ in Equation 3.15, the threshold is set too high to let any future documents be delivered, hence no learning takes place. Introducing a minimum delivery ratio (i.e., forcing the system to deliver at least a certain percentage of documents) guarantees enough training data to enable the system to recover automatically.

[6] Because the beta distribution is a conjugate prior for binomial distributions, we use it as a prior to simplify calculations.
[7] This is a special case of inverse gamma distribution, which is the conjugate prior for the variance of the normal distribution.

		Run 1	Run 2	Run 3	Run 4
		Basic Score Modeling	Basic Score Modeling + Min. Delivery	ML	ML + Min. Delivery
OHSU topics	**T9U′ utility**	1.84	3.25	2.7	**8.17**
	Avg. docs. delivered per profile	3.83	9.65	5.73	**18.40**
	Precision	0.37	0.29	0.36	0.32
	Recall	0.036	0.080	0.052	**0.137**
MESH topics	**T9U′ utility**	1.89	4.28	2.44	**13.10**
	Avg. docs. delivered per profile	3.51	11.82	6.22	**27.91**
	Precision	**0.42**	0.39	0.40	0.34
	Recall	0.018	0.046	0.025	**0.068**

Table 3.3. **Utility of each filtering run. OHSUMED dataset.**

On the OHSUMED dataset, for both OHSU topics and MESH topics, using maximum likelihood estimation plus a minimum delivery ratio achieved the best result. Although profile updating was disabled while filtering, all of the runs on this dataset received a positive average utility. The results for Run 4 on OHSU topics are above average compared with other filtering systems in TREC9 adaptive filtering track. This indicates how effective the threshold-setting algorithm is, because the other filtering systems learned improved profiles while filtering, whereas all of our experiments used static filtering profiles.

All four algorithms appear to perform about equally on FT data (Table 3.4). One difference between the FT dataset and the OHSUMED dataset is the average number of relevant documents per profile in the testing set (Section 3.3.5.2). Most of the FT filtering profiles are not good profiles, which means it is almost impossible to find a threshold that achieves a positive utility without profile updating. Indeed, the baseline method (Runs2) sets thresholds so high for some profiles that no relevant documents are delivered, thus getting zero Precision and Recall on those profiles. Our algorithm (Run 4) does not increase the threshold that much, which is why the baseline and unbiased ML methods appear similar according to the utility metric yet different according to the Precision and Recall metrics.

When we compared the utilities of each OHSU topic on Run 4 and Run 2 (Table 3.3), we found that our ML method did especially well on several good profiles where the score distribution of relevant documents and non-relevant documents are good from the system point of view (Figure 3.12). For other user profiles, the difference between our method and the baseline algorithm is not large. Comparison at the topic level and comparison between the OHSUMED and FT corpora suggests that our method works much better for good profiles, while its performance on other profiles are similar to the baseline method.

		Run 1	**Run 2**	**Run 3**	**Run 4**
		Basic Score Modeling	Basic Score Modeling + Min. Delivery	ML	ML + Min. Delivery
Topics 351-400	T9U′ utility	**1.44**	-0.209	0.65	0.84
	Avg. docs. delivered per profile	9.58	10.44	9.05	**12.27**
	Precision	0.20	0.17	0.22	**0.26**
	Recall	0.161	0.167	0.15	**0.193**
	Number of Profiles with zero Precision	24	22	24	**10**

Table 3.4. **Utility of each filtering run. FT dataset.**

The baseline method delivered fewer OHSUMED documents, on average, than the unbiased ML method (Figure 3.13, Table 3.3, columns 2 and 4). The average number of documents delivered by the baseline method was close to the minimum delivery ratio, which is empirical justification for our previous analysis illustrated in Figure 3.9: the Gaussian mean estimated by the baseline method was higher than the actual mean, thus the threshold was set too high. The final results of the baseline method appear to be highly influenced by the minimum delivery ratio. This is not a problem for the maximum likelihood method, because thresholds based on unbiased estimates of score distributions are more accurate.

We have not focused on computational efficiency in this report, but computational efficiency is important in environments where the filtering system must keep pace with a high-speed document stream. Although the mathematics may look complex, the unbiased ML algorithm is computationally efficient. It takes about 21 minutes ("wall-clock" time) to filter 4 years of OHSUMED data for 64 OHSU profiles (Table 3.3, Run 4) on a 500 MHz Intel Pentium III processor with 256 MB of memory. This time includes all aspects of document filtering, including the setting and updating of dissemination thresholds while filtering.

3.3.5.4 Summary

We have developed an effective algorithm for setting dissemination thresholds while filtering documents. Using a Gaussian distribution for the scores of relevant documents and an exponential distribution for the scores of non-relevant documents, a threshold that maximizes a given utility measure can be derived based on the parameters of the distribution.

Due to the fact that only relevance judgments of delivered documents are available, the method of estimating score density parameters proposed by Arampatzis, et al. (2001) is biased and results in a threshold higher than optimal. We proposed an unbiased algorithm based on the maximum likelihood principle to jointly estimate (i) the parameters of the two density

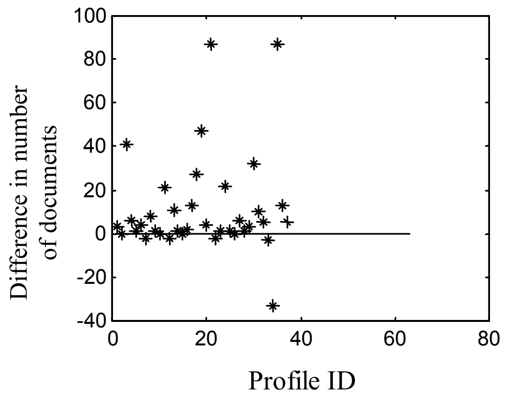


Figure 3.12. **Difference in number of documents delivered for OHSU topics: Run4 - Run 2. For most of the topics, Run 4 (ML) delivered more documents (indicated by points above the horizontal line) than Run 2 (baseline method).**

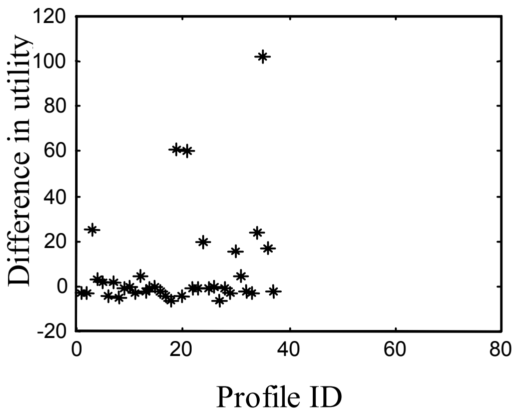


Figure 3.13. **Utility differences on OHSU topics: Run4 - Run2. For some topics, Run 4 (ML method) does especially better than Run 2 (baseline method) (indicated by points high above the horizontal line), while they work similar on other topics.**

distributions, and (ii) the ratios of relevant and non-relevant documents. The new algorithm explicitly models the inherent bias in the filtering environment, where training data consists only of documents with scores above a threshold that changes constantly. We believe this is the first report that solves the sample bias problem for information filtering. Our new method obtained significant improvements on the TREC-9 Filtering task.

In the experiments described above, filtering profiles were not updated while filtering. Although the effectiveness of the system is already very competitive, we believe combining threshold updating with profile updating will achieve better performance. One difficulty of combining our algorithm with profile updating is adjusting training data $D = \{(R_i, Score_i, \theta_i), i = 1 \Lambda\ N\}$, especially θ_i, based on new profile terms and weights. Future research can be focused on this problem. In the meantime, this new method would be expected to be superior in environments where profiles are updated less frequently, for example, weekly.

3.3.6 Assessment

It became clear early in the contract that an a good algorithm for learning dissemination thresholds would be crucial to a successful system. Four different methods were explored in the contract research: a heuristic method, a binning method, a method based on logistic regression, and a score modeling method. These algorithms were developed at different times during the contract, and were not compared under exactly the same experimental conditions, so it is difficult to quantify the differences among them. Nonetheless, certain qualitative differences are apparent.

Heuristic methods must be tuned carefully for a specific corpus and profiles with specific characteristics. When tuned appropriately, they can be very effective. However, in general they should be viewed as fragile.

Logistic regression is based on strong parametric assumptions about the form of the function that maps document scores to probabilities of relevance. In tests with relatively well-behaved profiles these assumptions were only true some portion of the time, perhaps 50-67% of the time. Although this approach is easy to understand, it should also be viewed as fragile. This approach is also somewhat complex computationally, so it is impractical to use it to modify thresholds each time user feedback is received.

The binning approach is robust and relatively effective. It, too, has some parameters (e.g., bin sizes), but its only assumption about the form of the function mapping document scores to probabilities is that the function is monotonically increasing, i.e., a higher score should not lead to a lower probability of relevance. Such a condition indicates a malformed profile in the extreme case, but small violations of this requirement are expected during filtering, and the binning algorithm must handle them. The main disadvantages of the binning method are that it requires more training data than other methods, and it responds a little slowly when the user changes the delivery requirements because it does not generalize. Otherwise it is effective and very efficient.

The score modeling approach is based on strong parametric assumptions about the form of the function that maps document scores to probabilities of relevance. In tests with relatively well-behaved profiles these assumptions were usually true. In tests with less well-behaved profiles, e.g., the profiles formed from Reuters categories, this algorithm performed poorly, and the reasons for its poor performance still aren't clear. The main advantages of this approach are that it requires relatively little training data, and it responds immediately when the user changes delivery requirements, because it generalizes well. The main disadvantage of this approach is that it is not yet integrated with profile updating, so when the profile changes (e.g., adding or deleting terms or adjusting term weights), the threshold learning must begin anew.

4. Filtering by Utility of Information (Novelty Detection)

A common complaint about information filtering systems is that they do not distinguish between documents that contain new relevant information and documents that contain information that is relevant but already known. An information filtering system would provide better service to its users if it identified three categories of documents for each user profile:

1. not relevant,
2. relevant but contains no new information, and
3. relevant and contains new information. Users could then decide for themselves how to treat relevant documents that contain no new information.

The decision about whether a document contains new information depends on whether the relevant information in the document is covered by information in documents delivered previously. This complicates the filtering problem. The relevance of a document is traditionally a stateless Boolean value. A document is or is not relevant, without regard to where the document appears in the stream of documents. Decisions about redundancy and novelty depend very much on where in the stream a document appears.

Novelty detection has been studied to some extent in recent Topic Detection and Tracking (TDT) research. However, our problem is somewhat different because it is set in a traditional information filtering environment. We are interested in delivering documents that are relevant and novel with respect to a known user profile; we also deliver documents that are relevant and redundant, but mark them as such so that the user can decide how they should be treated. Nonrelevant documents are discarded. The combination of an initial user profile and periodic interaction with the user also provides more information for learning than in TDT tasks.

For this study we defined a task and created an evaluation dataset that contains known redundant documents. We explicitly model relevance using a traditional adaptive filtering approach, and also explicitly model the information known by the user at a particular point in time. We model relevance and redundancy separately, and use quite different similarity measures for relevancy and redundancy. We also developed and tested a variety of redundancy measures. Since redundancy detection while filtering is a new research topic, we developed and tested a variety of algorithms for redundancy detection, including set distance, geometric distance (cosine similarity), and distributional distance (KL-divergence of language models).

4.1 Redundancy/Novelty Detection

We want our filtering system to distinguish among relevant documents that contain new (*novel*) relevant information and relevant documents that don't. When a document arrives, the system must determine whether it is on topic (relevancy detection), and if it is on topic, whether it is redundant (redundancy detection). We define "on topic"' as meaning that the document contains relevant information, as is traditional in information retrieval. We define "Redundant" to mean

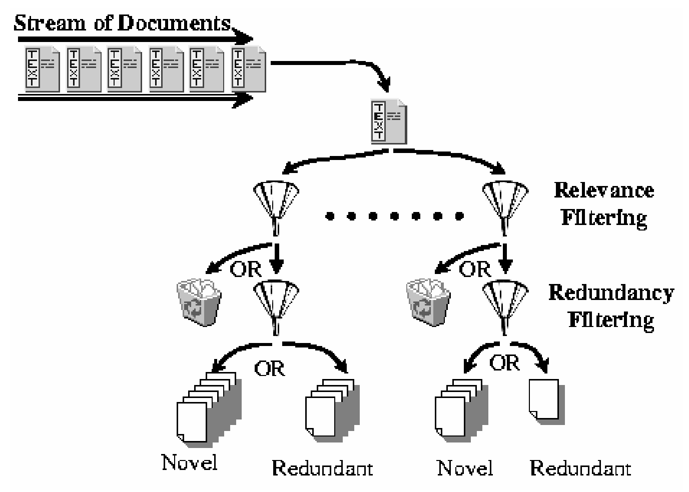


Figure 4.1. **A system that includes traditional document filtering (for relevance) as well as second stage novelty/redundancy detection.**

that all of the relevant information in the document is covered by relevant documents delivered to the user previously.[8] This definition of redundancy includes "duplicate" and "near duplicate" documents as well as documents that are redundant in content but very different in presentation. We assume that the information filtering system keeps track of which documents have been delivered to the user, and we assume, for simplicity, that the user views each document delivered.

The task of identifying novel and redundant documents has not been addressed by prior work, because of the lack of a clear definition of redundancy, and a lack of evaluation data. In the research reported here, novelty and redundancy are defined i) over the set of relevant documents, ii) with respect to previously seen documents, and iii) as opposite endpoints of a scale. The latter point is particularly important. When we treat novelty and redundancy as Boolean values, we imply a thresholding process that maps a value on a continuous redundancy/novelty scale to a Boolean value. We tested our approach to novelty and redundancy by creating an evaluation dataset judged by undergraduate assessors.

Our experience suggests that relevance and redundancy are significantly different concepts that require different solutions. A system that delivers documents that are novel and relevant must identify documents that are similar to previously delivered relevant documents in the sense of having the same topic, but also dissimilar to the previously delivered documents in the sense of containing new information. If the task is to deliver relevant documents, the learning algorithm will try to recognize documents similar to delivered relevant documents (training data). Indeed,

[8] Although "duplicate" and "near duplicate" documents are included in this definition of redundancy, the definition also encompasses documents that are redundant in content but very different in presentation.

traditional evaluation of filtering systems (e.g., the TREC Adaptive Filtering track (Robertson and Soberoff, 2001) actually rewards systems for delivering redundant documents. If the task is to deliver only documents containing novel information, the learning algorithm must avoid documents that are similar to those already delivered. These two goals are contradictory, and it may be unrealistic to expect a single component to satisfy them both.

This observation suggests a two stage approach to the problem, as shown in Figure 4.1. Traditional adaptive information filtering solutions can be used for relevancy filtering. It is less clear what type of algorithm should be used for redundancy filtering, and we defer discussion of solutions to Section 4.2 for now, we simply observe that a two-stage architecture is likely to simplify the problem.

We use the following notation throughout the report. All notation is defined with respect to a particular user profile.

- A, B: sets of documents.
- d_t: a document that arrives at time t and that is being evaluated for redundancy.
- D(t): the set of all documents delivered for the profile by the time d_t arrives, not including d_t.
- DR(t): the set of all relevant documents delivered for the profile. $DR(t) \subseteq D(t)$.
- $R(d_t)$: the measure of redundancy for document d_t
- d_i: usually refers to a relevant document that was delivered before d_t arrived.

The task was formulated based on the following assumptions, and they are basis for our research when acquiring redundancy judgements and developing algorithms.

Assumption 1: The redundancy of a new document d_t depends on D(t), the documents the user saw before d_t arrived. We use $R(d_t)=R(d_t \mid D(t))$ to measure this.

Assumption 2: $R(d_t \mid D(t))$ depends on all the *relevant* documents DR(t) the user has seen when d_t arrives, so $R(d_t \mid D(t)) = R(d_t \mid DR(t))$.

Assumption 3: For two documents set A, and B, if $B \subseteq A$ and B makes d_t redundant, then A also makes d_t redundant. To make it a softer assumption: $B \subseteq A \rightarrow R(d_t \mid A) \geq R(d_t \mid B)$.

Information filtering systems based on statistical retrieval models usually compute a numeric score indicating how well each document matches a profile; documents with scores above a profile-specific dissemination threshold are delivered. Similarly, the task of identifying redundant documents can be divided into two subtasks: calculate a score to measure how redundant each document is for a profile, then identify documents with scores above a profile-specific redundancy threshold. In our architecture (Figure 4.1), the second stage redundancy filter consists of two elements: i) redundancy score calculation, and ii) redundancy threshold learning.

In an adaptive filtering system, each of these architectural components defines a different research agenda. The scoring mechanism requires profile-specific “anytime” updating of redundancy measures. The threshold mechanism requires a threshold updating module. The former is the focus of the research described in this section. Although thresholding is not the focus of this section, we did implement a simple threshold setting algorithm to make the system more complete, and to enable evaluation of redundancy measures in the context of an operational filtering system.

4.2 Redundancy Measures

We assume that traditional information filtering techniques are used to identify relevant documents; we recognize that the filtering system will make mistakes, i.e., it will deliver some documents that are not relevant and discard some documents that are actually relevant. However, for simplicity we assume that novelty/redundancy detection is performed on a stream of documents that are presumed to be relevant. We frame this problem as finding a measure $R(d_t \mid DR(t))$, based on Assumption 2 in Section 4.1.

One approach to novelty/redundancy detection is to cluster all previously delivered documents, and then to measure the redundancy of the current document by its distance to each cluster. This approach is similar to solutions for the TDT First Story Detection problem. Our concerns about this approach are that it is sensitive to clustering accuracy, and is based on strong assumptions about the nature of redundancy, which we think is
user dependant.

Another approach is to measure redundancy based on the distance between the new document and each previously delivered document (document-document distance). This approach may be more robust than clustering, and may be a better match to how users view redundancy. When we asked assessors to annotate an evaluation dataset, we found that it was easiest for them to identify a new document as being redundant with a specific previously seen document, and harder to identify it as being redundant with *a set* of previously seen documents. This observation allows us to simplify the calculation of $R(d_t \mid DR(t))$ by setting it equal to the value of the maximally similar value in all $R(d_t \mid d_i)$.

$$R\left(d_t \mid DR(t)\right) = \operatorname{argmax}_{d_i \in DR(t)} R(d_t \mid d_i)$$

Although duplicate detection is not our goal, it is an instructive case because it is simple. If d_t and d_i are exact duplicates ($d_t = d_i$) then $R(d_t \mid d_i)$ should have a high value because a duplicate document is maximally redundant. One natural way to measure $R(d_t \mid d_i)$ is using measures of similarity/distance/difference between d_t and d_i .

Document timestamps are also an important source of evidence, because documents are more likely to be redundant with other recently delivered documents. During redundancy decisions truncating the delivery history to the most recent N documents delivered for a profile also reduces the number of documents that must be considered, which reduces computational costs. N is set to 10 in all experiments reported in this report. The experiments reported in this report considered only the 10 most recently delivered documents in redundancy comparisons.

One possibly subtle problem characteristic is that redundancy is not a symmetric metric. d_j may cause d_k to be viewed as redundant, but if the presentation order is reversed, d_k and d_j may both be viewed as containing novel information. A simple example is a document d_k that is a subset (e.g., a paragraph) of a longer document d_j. This problem characteristic motivates exploration of asymmetric forms of traditional similarity/distance/difference measures.

Below we present several different approaches to redundancy detection. The simple set distance measure is designed for a Boolean, set-oriented document representation. The geometric distance (cosine similarity) measure is a simple metric designed for "bag of words" document representations. Several variations of KL divergence and related smoothing algorithms are more complex metrics designed to measure differences in word distributions.

4.2.1 Set Difference

The set difference measure represents each document as a set of words. The novelty of a new document d_{t} is measured by the number of new words in the smoothed set representation of d_{t}. If a word w_{i} occurred frequently in document d_{t} but less frequently in an old document d_{i}, it is likely that new information not covered by d_{i} is covered by d_{t}.

Some words are expected to be frequent in a new document because they tend to be frequent in the corpus, or because they tend to be frequent in all relevant documents. "Stopwords" such as "the", "a", and "and" are examples of words that tend to be frequent in a corpus. There may also be topic-related stopwords, which are words that behave like stopwords in relevant documents, even if they are not stopwords in the corpus as a whole. An effective measure must compensate for both types of words.

Our set difference measure compensates for corpus stopwords by smoothing a new document's word frequencies with word counts from *all* previously seen documents. It compensates for topic stopwords by smoothing a new document's word frequencies with word counts from *all delivered* (presumed relevant) documents.

Thus we have the following measure for the redundancy of current document d_t with respect to old document d_i:

$$R(d_t \mid d_i) = \| \mathrm{Set}(d_t) \cap \overline{\mathrm{Set}(d_i)} \|$$

where:

$w_j \in \mathrm{Set}(d)$ iff Count $(w_j,d) > k$;
Count $(w_j,d) = \alpha_1 * tf_{w\,j,d} + \alpha_2 * df_{w\,j} + \alpha_3 * rdf_{w\,j}$;
$tf_{w\,j,d}$: the frequency of word w_j in document d;
$df_{w\,j}$: the number of filtered documents that contain w_j; and
$rdf_{w\,j}$: the number of times word w_j occurs in relevant documents the system has seen.

(α_1, α_2, α_3, k) are set to (0.8, 0.2, 0.0, 2) in our experiments; they could also be learned from training data.

We are not using the true difference between two sets:

$$\| \mathrm{Set}(d_t) \cap \overline{\mathrm{Set}(d_i)} \| + \| \overline{\mathrm{Set}(d_t)} \cap \mathrm{Set}(d_i) \|$$

here because the words in

$$\| \overline{\mathrm{Set}(d_t)} \cap \mathrm{Set}(d_i) \|$$

shouldn't contribute to the novelty of d_t and the optimal novelty measure should be asymmetric.

4.2.2 Geometric Distance

There are several different geometric distance measure, such as Manhattan distance and osine distance (Lee and Pereira, 1999). Since Manhattan distance is very sensitive to document length, Cosine distance maybe more appropriate for our task. Prior research showed that a Cosine distance based measure was useful for the TDT FSD task (Carbonell, et al., 2001).

Cosine distance is a symmetric measure related to the angle between two vectors (Jones and Furnas, 1987.) If we represent document d as a vector $d = (w_1(d), w_2(d), .., w_n(d))^T$, then:

$$\begin{aligned} R(d_t \mid d_i) &= \cos d_t, d_i \\ &= \frac{\sum_{k=1}^{n} w_k(d_t) w_k(d_i)}{\| d_t \| \ \| d_i \|} \end{aligned}$$

In our study, we used each unique word as one dimension, and set the tf.idf score as the weight of each dimension.

4.2.3 Distributional Similarity

Probabilistic language models have shown promise for identifying relevant documents in ad-hoc information retrieval tasks (e.g., Miller, et al., 2001; Kraaij, et al., 1999; Zhai and Lafferty, 2001). Their strong theoretical foundation supports a variety of new capabilities, including redundancy detection. In the language model approach, a document d is represented by a unigram word distribution Θ_d. Kullback-Leibler divergence, a distributional similarity measure, is one way to measure the redundancy of one document given another.

$$\begin{aligned} R(d_t \mid d_i) &= -KL(\Theta_{d_t}, \Theta_{d_i}) \\ &= -\sum_{w_i} P(w_i \mid d_t) \log\left(\frac{P(w_i \mid d_i)}{P(w_i \mid d_t)}\right) \end{aligned}$$

where Θ_d is the language model for document d, and is a multinomial distribution.

Θ_d can be found by maximum likelihood estimation (MLE):

$$P(w_i \mid d) = \frac{tf(w_i, d)}{\sum_{w_j} tf(w_i, d)}$$

The problem with using MLE is that if a word never occurs in document d, it will get a zero probability ($P(w_i|d)=0$). Thus a word in d_t but not in d_i will make $KL(\Theta_{dt}, \Theta_{di}) = \infty$.

Smoothing techniques are necessary to adjust the maximum likelihood estimation so that the KL-based measure is more appropriate. Prior research shows that retrieval performance is highly sensitive to smoothing parameters. Several smoothing methods have been applied to ad-hoc information retrieval and text classification (e.g., (Zhai and Lafferty, 2001b; McCallum, et al., 1998)). Based on this prior research, we selected two methods: Bayesian smoothing using Dirichlet priors, and shrinkage.

4.2.3.1 Bayesian Smoothing Using Dirichlet Priors

This approach to smoothing uses the conjugate prior for a multinomial distribution, which is the Dirichlet distribution (Zhai and Lafferty, 2001b). For a Dirichlet distribution with parameters $\lambda\, p(w_1)$, $\lambda\, p(w_2)$,..., $\lambda\, p(w_n)$ the posterior distribution using Bayesian analysis for Θ_d is:

$$P_\lambda(w_i \mid d) = \frac{tf(w_i, d) + \lambda p(w_i, d)}{\sum_{w_j} \left(tf(w_j, d) + \lambda p(w_j, d)\right)}$$

In our experiments, if w_j is in d_t, we set $\lambda\, p(w_j)=0.5$, otherwise $\lambda\, p(w_j)=0$.

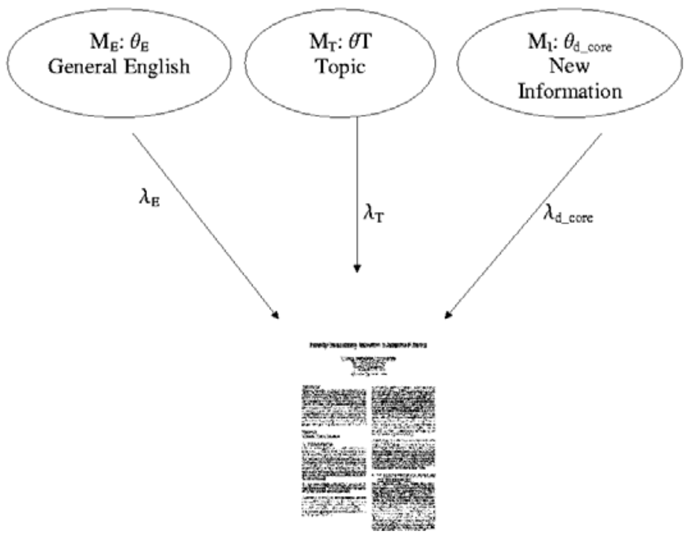


Figure 4.2. **A mixture model for generating relevant documents.**

4.2.3.2 Smoothing Using Shrinkage

This approach smooths by "shrinking" parameter estimates in sparse data towards the estimates in rich data (McCallum, et al., 1998). This is a special case of the more general Jelinek-Mercer smoothing method, which involves deleted-interpolation estimation of linearly interpolated n-gram models (Zhai and Lafferty, 2001b). For estimating the language model of document d, we can shrink its MLE estimator LM_{d_MLE} with the MLE estimator of a language model for general English Θ_{E_MLE} and the MLE estimator of a language model for the topic Θ_{T_MLE}:

$$\Theta_d = \lambda_d \Theta_{d\_MLE} + \lambda_T \Theta_{T\_MLE} + \lambda_E \Theta_{E\_MLE} \tag{4.1}$$

where $\lambda_d + \lambda_T + \lambda_E = 1$.

λ_E can be estimated from the documents the filtering system has processed, and λ_T can be estimated from the documents the filtering system has delivered (presumed relevant documents). We can derive empirical optimal values for λ_d, λ_T, and λ_E using "leave-one-out" cross validation as described in (McCallum, 1998). In our experiment, we used "leave-0.5-out".

4.2.4 A Mixture Model

In this section we introduce a new algorithm based on a generative model of document creation. This approach uses probabilistic language models and KL distance as described above. However, this new mixture model measure is based on a novel view of how relevant documents are generated. We can also view it as a language model with a smoothing algorithm designed specifically for our task.

As shown in Figure 4.2, we assume each relevant document is generated by the mixture of three language models: A General English language model Θ_E, a user-specific Topic Model Θ_T, and a document-specific Information Model Θ_{d_core}. Each word w_i in the document is generated by each of the three language models with probability λ_E, λ_T and λ_{d_core} respectively:

$$P(w_i \mid \Theta_E, \Theta_T, \Theta_{d\_core}, \lambda_E, \lambda_T, \lambda_{d\_core}) = \lambda_E P(w_i \mid \Theta_E) + \lambda_T P(w_i \mid \Theta_T) + \lambda_{d\_core} P(w_i \mid \Theta_{d\_core}) \qquad (4.2)$$

where $\lambda_E + \lambda_T + \lambda_{d\_core} = 1$.

For example, if information need is “Star War”', in a relevant document words such as “is” and “the” probably come from the general English model Θ_E. Words such as “star” and “wars” probably come from the Topic Model Θ_T. For a document with the title “Martin Marietta Is Given Contract For Star Wars Site”, the words “Martin” and ”Marietta” are more likely to be generated from the new information model Θ_{d_core}. Θ_T is the core information of a topic, while Θ_{d_core} is the core information of a particular relevant document. When a filtering system uses relevance feedback to improve the user profile, it focuses on learning Θ_T to find words that are correlated with relevance. When doing redundancy detection, the filtering system must focus on finding the document information model Θ_{d_core} that describes new information covered by a document d. A document with a Θ_{d_core} that is quite different from that of documents the user has read before is more likely to be novel. Thus a mixture model similarity measure between two documents based on the core information contained in each document can be defined.

$$R(d_t \mid d_i) = KL\left(\Theta_{d_t\_core}, \Theta_{d_i\_core}\right)$$

If we fix λ_E, λ_T, and λ_{d_core} then there exists a unique optimal value for the document core model $\Theta^*_{d\_core}$ that maximizes the likelihood of the document.

$$\Theta^*_{d\_core} = \operatorname{argmax}_{\Theta_d} P\left(d \mid \Theta_E, \Theta_T, \Theta_{d\_core}, \lambda_E, \lambda_T, \lambda_{d\_core}\right)$$

Three points are worth noting about the mixture model.

- Although Equations 4.1 and 4.2 look similar, the computations performed, and the final model acquired and used to calculate KL divergence, are quite different. In Equation 4.1, all three language models on the right side of the formula are calculated by counting; the interpolation weights λ_E, λ_T, and λ_d are the parameters that must be computed. By contrast, Equation 4.2 computes the document information model Θ_{d_core}. Equation 4.1 uses shrinkage to increase the probability of words that occur frequently in the topic or in general English if they occur less frequently in document d. Equation 4.2 uses a mixture model to decrease the probability of these words. Intuitively, those words are not the core information of a document, and the difference between two documents d_t and d_i due to them does not help in a redundancy decision. By shrinking with Θ_{E_MLE} and Θ_{T_MLE}, shrinkage smoothing reduces the distance between two documents due to those words, thus reducing their effect on the redundancy measure. With the mixture model, we directly decrease the probability of those words to reduce their effect.

- We must fix the values of λ_E, λ_T, and λ_{d_core}. If we train λ's and Θ_{d_core} together, we get $\lambda_{d\_core} = 1$ and $\Theta_{d\_core} = \Theta_{d\_MLE}$, which is the unsmoothed language model for document d; the benefit of smoothing, which decreases the effects of uninformative words, is lost.

- This model intentionally focus on *"what's new"* in a document, thus it avoids the contradiction between identifying relevance and novelty. It is inherently difficult to use existing algorithms to identify both relevant and redundant documents, because the two tasks require different approaches. If the task is to deliver relevant documents, the learning algorithm will try to recognize documents similar to already delivered relevant documents (training data). If the task is to deliver only documents containing novel information, the learning algorithm must avoid documents that are similar to those already delivered. This model introduces Θ_T and Θ_{d_core}, which means the measure of relevancy and redundancy are focused on different parts of a document. For relevancy, the system would like to focus on Θ_T, while for redundancy it should focus on Θ_{d_core}. Thus the two tasks are no longer contradictory

We can train Θ_T , Θ_{d_score}, and Θ_E using the EM algorithm, which has been used by others to find a language model for similar problems (e.g., (Kraaij, et al., 1999; Zhai and Lafferty, 2001b)).

4.3 Redundancy Thresholds

When we observed human assessors making redundancy decisions, we found that two annotators working on the same topics sometimes disagreed. Sometimes the disagreement was due to differences in the assessors' internal definition of redundancy. More often one assessor might feel that a document d_t should be considered redundant if a previously seen document d_i covered 80% of d_t; the other assessor might not consider it redundant unless the coverage exceeded 95%.

A person's tolerance for redundancy can be modeled with a user-dependent threshold that converts a redundancy score into a redundancy decision. User feedback about which documents are redundant serves as training data. Over time the system can learn to estimate the probability that a new document with a given redundancy score would be considered redundant : P(user j thinks d_t is redundant | R (d_t | DR(t))).

This two-step process first maps the document d_t into a 1-dimensional space $R(d_t \mid DR(t)$ using a redundancy measure, and then learns from training data the probability of redundancy given the value on this dimension. This approach is similar in spirit to how many adaptive filtering systems identify relevant documents (e.g., (Robertson, 2000; Zhang and Callan, 2001)).

Ideally, an optimization goal should be set before deciding what kind of threshold setting algorithm to use. However, our first step is a very simple algorithm for setting thresholds. Our solution is intentionally simple, in part because of the lack of adequate test collection labeled with redundant information for a given profile, and in part because this problem is not (yet) our research focus.

The algorithm for learning redundancy thresholds is:

- Initialize redundancy threshold RThreshold to a value that is so high that only very redundant documents (e.g., near duplicates) are considered redundant; and
- For each document d_t delivered (which means $R(d_t) <$ RThreshold when d_t arrives), ask the user if the document is redundant.
- If the document is redundant and if $R(d_t) > R(d_i)$ for all $d_i \in DR(t)$
  then Rthreshold = $R(d_t)$
  else RThreshold = Rthreshold – (Rthreshold – $R(d_t)$) / 10.

This is clearly a weak algorithm, because it only decreases the threshold. If the threshold becomes too low there is no method of increasing it again. The effectiveness of this algorithm is explored in Section 4.1.1.5.

4.4 Experimental Methodology

4.4.1 AP News & Wall Street Journal Dataset

We created a one gigabyte dataset by combining AP News and Wall Street Journal data from TREC CDs 1, 2, and 3. We chose these corpora because they are widely available, because information needs and relevance judgements are available from NIST, and because the two newswire corpora cover the same time period (1988 to 1990) and many of the same topics, guaranteeing some redundancy in the document stream. Documents were ordered chronologically. 50 TREC topics (101 to 150) simulated user profiles.

The decision about how to collect redundancy assessments depends in part upon how we view the task. If we viewed redundancy as a relationship between document d_t and a set of documents, for example a subset of the documents delivered for a particular profile, it would be impossible to collect redundancy assessments. We would need to enumerate all of the possible subsets of documents delivered at time t and then ask assessors to judge whether d_t is redundant with respect to each set. The number of possible subsets is 2^{t-1}, which is impractical for all but very small values of t. Although we know that in the "real world" redundancy is based on the set of documents delivered previously, we can only model it as a relationship among pairs of documents. This is the approach we adopted when developing algorithms, but that decision was based in part on how we intended to collect redundancy judgements.

We hired undergraduate students, who were otherwise unaffiliated with our research, to read the relevant documents for a profile in chronological order and to provide redundancy judgments. The decision to restrict their attention to relevant documents is based on assumption 2 in Section 4.1, and is consistent with a filtering system where another component makes decisions about relevance.

Assessors judged one topic at a time. They were instructed to make a decision for each document about whether the information it contained was redundant with document(s) seen previously for that topic, and to identify the prior document(s). Each topic was judged by two assessors and then differences were resolved by the assessors themselves.

We believe that in operational environments different people will have different definitions of redundancy and different redundancy thresholds. We modelled this environment by not giving assessors a precise definition of redundance. We provided two degrees of redundancy, *absolutely redundant* and *somewhat redundant*; assessors could apply them based on their expectations about how a good system should behave. If the assessor thought a person would definitely not want to read d_t because it absolutely contained no new information, d_t was marked as *absolutely redundant*. If the assessor thought that a new document had some new information that a person might want to read, even though much of the document was redundant with a prior document, the document could be marked as *somewhat redundant*. Documents that were not *completely redundant* or *somewhat redundant* were marked as *novel*.

An example of the redundancy assessments is shown below. The first field is a profile id. The second field is the document id of a redundant document. Subsequent document ids are the documents that preceded it in the stream and that made it redundant. A `?' indicates that a document is only partially redundant.

- **q121 AP880214-0049 ? AP880214-0002**
  if user q121 read document AP880214-0002, then AP880214-0049 is somewhat redundant.
- **q121 AP880217-0031 AP880216-0137**
  if user q121 read document AP880216-0137, then AP880217-0031 is absolutely redundant.
- **q128 AP880218-0137 AP880218-0113 AP880218-0112**
  if user q128 read AP880218-0113 and AP880218-0112, then AP880218-0137 is absolutely redundant.

On average there are about 66.5 records per TREC topic (note that a single record may relate a document to several prior documents). About 19.2 records per profile are *absolutely redundant*; the rest represent partial redundance.

Students reported that the choice of corpus (“old”) and topics made this a dull task, so we were unable to collect assessments for all 50 topics. The results in this report are based on a set of adjudicated assessments for 33 profiles.

4.4.2 TREC Interactive Dataset

We combined the dataset used by the TREC-6, TREC-7, and TREC-8 Interactive Tracks to create a test dataset containing 210,158 documents from the 1991-1994 Financial Times of London. There are 20 TREC topics; each one defines a user profile. For each topic, TREC assessors have identified several instances. Different instances are about different aspects of the topic. A document on a topic could be mapped to multiple instances of that topic. The mapping from the relevant documents to instances is provided by NIST. In our evaluation, we treated each instance as one aspect of the topic and assumed that a user only wants to see one document on each aspect. Thus a document is redundant if the user has already seen at least one document for each instance this new document belongs to.

Since this dataset was not created explicitly for redundancy detection, it maybe not be as well-matched to the task as the AP News/Wall Street Journal dataset described above. However, we felt that a second dataset, even one that isn't perfect, would be a useful source of information.

4.4.3 Evaluation Methodology

We believe that it is important to evaluate a particular component of a system with a metric that is not affected by strengths and weaknesses in other parts of system. In this case, we would like to factor out how well the filtering system identifies relevant documents and sets redundancy thresholds. In our experiments we assume that the filtering system identifies relevant documents with 100% precision and recall by evaluating redundancy filtering only on a stream of documents marked relevant by NIST assessors. In some tests we also evaluate the effectiveness of redundancy-scoring algorithms, and factor out the effect of the redundancy threshold algorithm, by reporting average Precision and Recall figures *for redundant documents*. Precision and recall are well-known metrics in IR community. We adapt them to the redundancy detection task as shown below.

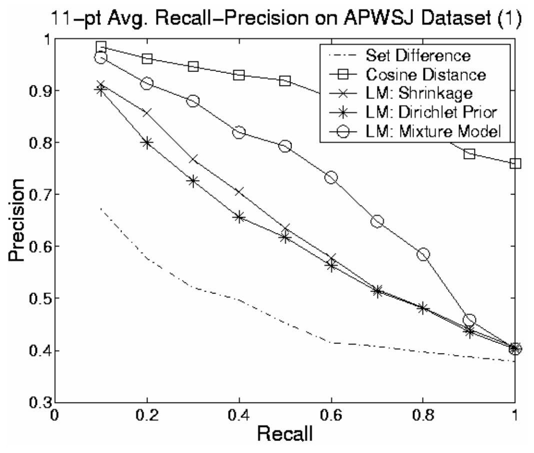


Figure 4.3. **Comparing redundancy measures on AP News & Wall Street Journal data. A document is considered *redundant* if an assessor marked it *absolutely redundant* or *somewhat redundant*.**

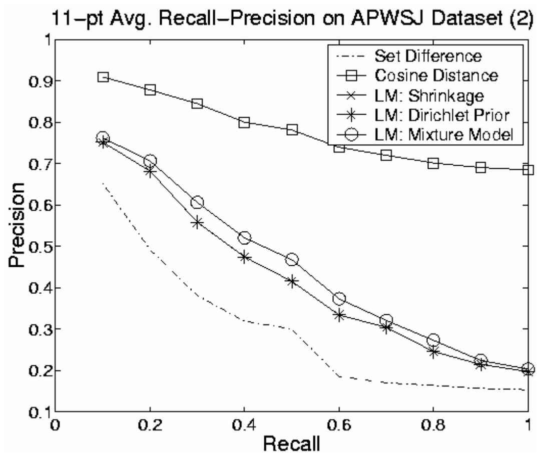


Figure 4.4. **Comparing redundancy measures on AP News & Wall Street Journal data. A document is considered *redundant* only if an assessor marked it as *absolutely redundant*. The Language Model (LM) measures using Shrinkage and Dirichlet Prior smoothing perform equally, thus overlap.**

$$\begin{aligned}
\text{Redundancy Precision} &= \frac{R^-}{R^- + N^-} \\
\text{Redundancy Recall} &= \frac{R^-}{R^- + R^+} \\
\text{Redundancy Mistake} &= \frac{R^+ + N^-}{R^+ + N^- + R^- + N^+}
\end{aligned}$$

R^-, R^+, N^-, N^+ correspond to the number of documents that fall into the following categories

	Redundant	Non-Redundant
Delivered	R^+	N^+
Not Delivered	R^-	N^-

For simplicity, we will use *Precision* and *Recall* to refer to *Redundancy Precision* and *Redundancy-Recall* in the rest of this report.

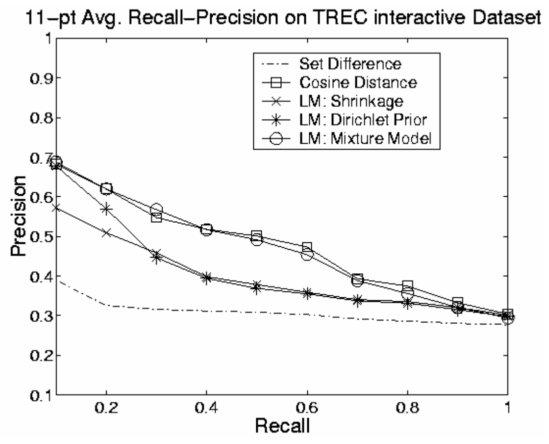


Figure 4.5. **Comparing redundancy measure on TREC Interactive Track data. A document about aspect(s) already covered by previously delivered documents is considered redundant.**

4.5 Experimental Results

The five redundancy measures described in Section 4.2 were compared on the two datasets described in Sections 4.1.1.4. A redundancy score was calculated for each relevant document d_t, based on the relevant documents d_i that preceded it in the document stream. The results are shown in Figures 4.3, 4.4, and 4.5 in the form of average Recall-Precision graphs over the set of redundant documents.

On both datasets the Set Difference measure is the least accurate. Representing a document as a set of Boolean word features, even with smoothing to add or delete additional words, was very ineffective.

The traditional cosine-similarity metric (a geometric distance measure) was very effective. This result was a small surprise, because cosine similarity is less well-justified theoretically than the language modelling approaches. The cosine similarity metric is also symmetric; we expected asymmetric measures to be a better model of this task. However, cosine similarity has been demonstrated many times and over many tasks to be a robust similarity metric. Our results add redundancy detection to the long list of tasks for which it is effective.

The results for the three Language Modelling algorithms confirm prior research showing the importance of selecting a good smoothing algorithm. The mixture model approach was consistently more accurate than the other two smoothing algorithms on both corpora. It was also about as effective as the cosine similarity measure on the TREC Interactive Track dataset. This approach to mixing information from corpus, topic, and a document language models provides a new point of view about how to model documents, and it does deliver improved effectiveness compared to other language modelling approaches. This result is not completely surprising,

Measure	Recall	Precision	Mistake
Set Distance	0.52	0.44	43.5%
Cosine Distance	0.62	0.63	28.1%
LM: Shrinkage	0.80	0.45	44.3%
LM: Dirichlet Prior	0.76	0.47	42.4%
LM: Mixture Model	0.56	0.67	27.4%

Table 4.1. **Average performance of different redundancy measures with a simple thresholding algorithm, measured on 33 topics with the AP News & Wall Street Journal dataset. Both *absolutely redundant* and *somewhat redundant* documents are treated as *redundant*.**

Measure	Recall	Precision	Mistake
Set Distance	0.36	0.29	28.1%
Cosine Distance	0.48	0.33	18.7%
LM: Shrinkage	0.375	0.45	21.0%
LM: Dirichlet Prior	0.375	0.45	21.0%
LM: Mixture Model	0.46	0.40	16.7%

Table 4.2. **Average performance of different redundancy measures with a simple thresholding algorithm, measured on 33 topics with the AP News & Wall Street Journal dataset. Only *absolutely redundant* documents are treated as *redundant*.**

because the algorithm explicitly models *what's new* in a document. However, these results suggest that it is not as robust as the cosine similarity measure.

We also implemented a simple threshold-setting algorithm (Section 4.1.1.3). The threshold-setting algorithm is simple and weak, in part because we did not set an optimization goal specifying the relative rewards and penalties for delivering novel and redundant documents. However, even a simple algorithm can be used to analyze the different redundancy measures. In particular, it provides a more accurate indication of what a user might see in an operational environment.

Tables 4.1, 4.2, and 4.3 summarize the effectiveness of the five redundancy measures when used with the simple redundancy threshold algorithm. Results are reported for both datasets. (The metrics are described in Section 4.1.1.4.)

If we evaluate the redundancy measures by the percentage of mistakes they make, the Cosine Similarity and Mixture Model redundancy measures are much better than the rest. These two measures yields a reasonably low percentage of mistakes when a strict definition of redundancy is used (Table 4.2), but a less satisfying percentage when *somewhat redundant* documents are

Measure	Recall	Precision	Mistake
Set Distance	0.43	0.28	46.8%
Cosine Distance	0.45	0.44	34.5%
LM: Shrinkage	0.79	0.33	53.3%
LM: Dirichlet Prior	0.73	0.34	49.0%
LM: Mixture Model	0.18	0.51	28.4%

Table 4.3. **Average performance of different redundancy measures with a simple thresholding algorithm, measured on 20 topics with the TREC Interactive dataset.**

treated as redundant (Table 4.1). This result implies that our simple redundancy threshold algorithm models the user by treating *somewhat redundant* as novel. We know that a good threshold setting algorithm is important to system accuracy, and we hypothesize that threshold setting should depends on each user. Table 4.2 shows that reasonably good accuracy is possible when the thresholding algorithm is well-matched to the task.

4.6 Assessment

The research reported here is a first step towards adaptive information filtering systems that learn to identify documents that are novel and redundant in addition to relevant and nonrelevant. It defines a task, an evaluation methodology, and a set of novelty/redundancy measures. A reusable corpus was created from generally available documents, a set of adjudicated redundancy judgements was created, and an existing corpus was adapted to our task.

The experimental results demonstrate that it is possible to identify redundant documents with reasonable accuracy. They also demonstrate the importance of a suitable redundancy-threshold algorithm, analogous to the relevance-threshold algorithm found in many information filtering systems. Our results also suggest that the algorithm itself should depend on the user model of redundancy. The extremely small amount of training data (less than what is available for relevance-based adaptive filtering) makes it a challenging problem.

Five measures were proposed for assessing the redundance of a new document with respect to a previously seen stream of documents. The experimental results demonstrate that the well-known cosine similarity metric is effective on this new task. They also demonstrate that a new metric based on a mixture of language models can be as effective as the cosine similarity metric in some cases.

We believe that the metric based on a mixture of language models is an important contribution, whether or not it was the most effective algorithm for this task. We believe that viewing documents as a mix of information covered by corpus, topic, and "new information" models is an appropriate model of an information filtering task. The results reported here are a first attempt to apply this approach to a realistic task; we expect to see other attempts in the future.

This research is only a first attempt at redundancy/novelty detection in an adaptive filtering environment, so there are many open problems for future research. Our research on profile-specific “anytime” updating of redundancy measures just scratches the surface. Although cosine similarity worked well in our experiments, we believe that the underlying redundancy relationship is asymmetric, and that asymmetric measures will eventually be more accurate. It is also likely that other features, such as timestamp, document source, phrases, and proper names will be important sources of evidence for novelty decisions.

Our research measured redundancy based on document-document pairs, which is easy to assess and easy to model, but the underlying task is probably better modelled by comparing clusters of delivered documents to the new document. The best choice may be problem-specific, e.g., depending upon corpus or profile characteristics.

5. Prototype System

Some of the experiments described in this report, particularly the early work on dissemination thresholds, were done with InRoute, an adaptive information filtering system written at the University of Massachusetts. Licensing restrictions made InRoute difficult to modify and distribute, so a new information filtering system was developed "off contract". The new system, called YFilter, was functionally similar to InRoute, but had a simpler and "cleaner" architecture. Most of the experiments described in this report were done using the YFilter adaptive information filtering system.

Generally, an adaptive filtering system that can learn from user feedback has several important subtasks:

1. Learning corpus statistics, such as the inverse document frequency (*idf*) of each term;
2. Learning filtering profiles that describe desired documents, which usually means adding or deleting terms and adjusting term weights;
3. Measuring the redundancy between new documents and documents delivered previously; and
4. Learning dissemination thresholds that specify how closely a document must match a profile and how novel it must be in order to be delivered.

YFilter does all of the above tasks. Given a corpus file that contains a sequence of incoming documents and a profile file that contains initial user profiles, YFilter can go through the corpus, compare each document with each profile and deliver the most relevant and novel documents to each user. YFilter can learn from user relevance feedback while filtering, updating profile descriptions and thresholds based upon user-supplied relevance judgements. YFilter can also learn a person's novelty criteria.

The core filtering engine is written in C++. YFilter was designed as a library with an application programmer's interface (API), so that software engineers can embed it in a variety of application environments. It is delivered with two user interfaces: i) a graphical user interface (GUI) written in Java, and ii) a UNIX-style command-line interface suitable for use in scripts and batch files. YFilter runs under Windows NT and the Sun Solaris version of Unix.

The YFilter installation procedure, file formats, graphical user interface, batch interface, and example runs are described in the *YFilter User Manual*. The YFilter API and internal design is described in the *YFilter System Design Document*. Please see those documents for more information on the prototype system.

YFilter, its documentation, and sample data files were distributed to the Air Force Research Laboratory on CD-ROM. A copy is available from the Project Engineer.

6. Lessons Learned

A variety of topics were explored during the contract research, and a variety of lessons were learned. This section summarizes them briefly.

The Rocchio algorithm was the most consistently effective machine learning algorithm for learning profiles. Rocchio was tested across a variety of document streams, types of profiles, and learning conditions. It was always the most effective. This result is consistent with the research findings of other groups during the same period. The Rocchio algorithm is also computationally efficient, so it is quite compatible with online adaptive filtering systems. Most of the research reported here, including tests with profile sets of up to 5,000 profiles, was done on ordinary Intel-based personal computers running Windows NT.

Three techniques were explored to improve the expressive power of profiles learned incrementally during adaptive filtering: use of phrases, use of named-entities, and use of hierarchically organized profiles. All three were expected to work well, and have delivered significant improvements in accuracy in Routing or batch filtering environments. However, none of them were particularly effective in the adaptive filtering experiments conducted under this contract.

One problem with the research conducted under the contract was that it was restricted primarily to TREC Filtering data and the TREC filtering methodology, which may not be represent the conditions in government environments. In particular, the TREC Filtering tracks have recently focused on use of very small amounts of training data and a relatively sparse stream of relevant documents, i.e., the needle in the haystack type of problem. The lack of sufficient training data is clearly an obstacle to assessing the effectiveness of complex features, such as phrases and co-occurrence relationships. Complex features are more precise than single terms, but they are also occur more rarely, so there are fewer opportunities to learn appropriate term weights. It is likely that some of the techniques developed during the contract research would be more effective in environments that have more than a few positive examples in a 1-4 year document stream.

Likewise, the Reuters 2001 corpus is probably not a good model for the use of hierarchical filtering profiles in government environments. The use of hierarchical profiles clearly improved the amount of training data (the number of relevant and non-relevant documents) for some Reuters categories. However, the Reuters 2001 categories are not a good model of information needs in an adaptive filtering environment, so the it isn't possible to draw firm conclusions about the effectiveness of hierarchical profiles.

Four different methods of learning dissemination thresholds were explored in the contract research: a heuristic method, a binning method, a method based on logistic regression, and a score modeling method. The heuristic and logistic regression methods are based on strong parametric assumptions about the form of the function that maps document scores to probabilities of relevance. In tests with relatively well-behaved profiles these assumptions were only true some portion of the time. Although these approaches are easy to understand, we consider them fragile.

The binning approach is robust and relatively effective. It, too, has some parameters (e.g., bin sizes), but its only assumption about the form of the function mapping document scores to probabilities is that the function is monotonically increasing, i.e., a higher score should not lead to a lower probability of relevance. Such a condition indicates a malformed profile in the extreme case, but small violations of this requirement are expected during filtering, and the binning algorithm must handle them. The main disadvantages of the binning method is that it requires more training data than other methods, and it responds a little slowly when the user changes the delivery requirements because it does not generalize.

The score modeling approach is based on strong parametric assumptions about the form of the function that maps document scores to probabilities of relevance. In tests with relatively well-behaved profiles these assumptions were usually true. In tests with less well-behaved profiles, e.g., the profiles formed from Reuters categories, this algorithm performed poorly, and the reasons for its poor performance still aren't clear. The main advantage of this approach is that it requires relatively little training data, and it responds immediately when the user changes delivery requirements, because it generalizes well. The main disadvantage of this approach is that it is not yet integrated with profile updating, so when the profile changes (e.g., adding or deleting terms or adjusting term weights), the threshold learning must begin anew.

The research reported here included a first step towards adaptive information filtering systems that learn to identify documents that are novel and redundant in addition to relevant and nonrelevant. It defined a task, an evaluation methodology, and a set of novelty/redundancy measures. A reusable corpus was created from generally available documents, a set of adjudicated redundancy judgements was created, and an existing corpus was adapted to our task.

The experimental results demonstrate that it is possible to identify redundant documents with some accuracy. Five measures were proposed for assessing the redundancy of a new document with respect to a previously seen stream of documents. The experimental results demonstrate that the well-known cosine similarity metric is effective on this new task. They also demonstrate that a new metric based on a mixture of language models can be as effective as the cosine similarity metric in some cases.

We believe that the metric based on a mixture of language models is an important contribution, whether or not it was the most effective algorithm for this task. We believe that viewing documents as a mix of information covered by corpus, topic, and "new information" models is an appropriate model of an information filtering task. The results reported here are a first attempt to apply this approach to a realistic task; we expect to see other attempts in the future.

Although cosine similarity worked well in our experiments, we believe that the underlying redundancy relationship is asymmetric, and that asymmetric measures will eventually be more accurate. It is also likely that other features, such as timestamp, document source, phrases, and proper names will be important sources of evidence for novelty decisions.

7. Publications

The following publications were produced under the contract.

1. J. Allan, J. Callan, F. Feng, and D. Malin. (2000.) "INQUERY and TREC-8." In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. National Institute of Standards and Technology, special publication 500-246.

2. S. Luo and J. Callan. (2001.) "Named entity detection." Technical Report, Language Technologies Institute, Carnegie Mellon University. February, 2001.

3. Y. Zhang and J. Callan. (2001.) "A generative model for filtering thresholds." *Workshop on Language Modeling and Information Retrieval*, Carnegie Mellon University, May 31-June 1, 2001.

4. Y. Zhang and J. Callan. (2001.) "Maximum likelihood estimation of filtering thresholds." In *Proceedings of the Twenty Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.

5. Y. Zhang and J. Callan. (2001.) "YFilter at TREC-9." In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology, special publication 500-249.

6. Y. Zhang and J. Callan. (2002). "The bias problem and language models in adaptive filtering." In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*. National Institute of Standards and Technology, special publication 500-250.

7. Y. Zhang, J. Callan, and T. Minka. (2002.) "Novelty and redundancy detection in adaptive filtering." In *Proceedings of the Twenty Fifth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Tampere, Finland: ACM. **Best paper award.**

8. Y. Zhang, W. Xu., and J. Callan. (2002.) "Exact maximum likelihood estimation for word mixtures." In *ICML 2002 Workshop on Text Learning (TextML 2002).*

8. References

1. J. Allan (ed). (2002.) *Topic detection and tracking: Event-based information organization*. Kluwer Academic Publishers

2. J. Allan. (1996.) Incremental relevance feedback for information filtering. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 270-278. ACM.

3. J. Allan, J. Callan, F. Feng, and D. Malin. (2000.) "INQUERY and TREC-8." In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. National Institute of Standards and Technology, special publication 500-246.

4. A. Arampatzis, J. Beney, S. H. A. Koster, and T. P. van der Weide. (2001.) "Incrementality, half-life, and threshold optimization for adaptive information filtering. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology, special publication 500-249.

5. A. Arampatzis and A. van Hameren. (2001.) "The score-distributional threshold optimization for adaptive binary classification tasks." In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

6. T.Ault, and Y. Yang. (2001.) kNN at TREC-9: A failure analysis. In *Proceeding of Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology, Special Publication.

7. D. M. Bikel, S. Miller, R. Schwartz, and R. M. Weischedel. (1997.) "Nymble: A High-Performance Learning Name-finder. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pp. 194-201.

8. D. M. Bikel, R. Schwartz, and R. M. Weischedel. (1999.) "An Algorithm that Learns What's in a Name." Machine Learning Journal, vol 34, pp. 211-231.

9. J. Broglio, J.P. Callan, W.B. Croft, and D.W. Nachbar. (1995.) Document retrieval and routing using the INQUERY system. In *Proceeding of Third Text REtrieval Conference (TREC-3)*, pp. 29-38. National Institute of Standards and Technology, Special Publication 500-225.

10. C. Buckley, A. Singhal, M. Mitra, and G. Salton. (1995.) "New retrieval approaches using SMART." In *Proceedings of 1995 Text REtrieval Conference (TREC-3).* National Institute of Standards and Technology, special publication

11. C. Buckley, J. Allan, and G. Salton. (1994.) "Automatic routing and ad-hoc retrieval using SMART: TREC-2. In Proceedings of the Second Text REtrieval Conference (TREC-2). NIST Special Publication 500-215.

12. J. Callan. (1996.) Document filtering with inference networks. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 262-269. ACM.

13. J., Callan. (1998.) "Learning while filtering documents." In *Proceedings of the Twenty First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

14. J. Callan. (2000.) "System and method for filtering a document stream." U.S. Patent 6,105,023, August 15, 2000.

15. J. Callan, W.B. Croft, and J. Broglio. "TREC and TIPSTER experiments with INQUERY." *Information Processing and Management*, 31(3):327-343, 1995

16. J. Carbonell, Y. Yang, R. Brown, C. Jin, and Jian Zhang. (2001.) "CMU TDT Report 13-14 Nov 2001." *Topic Detection and Tracking Workshop Report*.

17. B. Croft, J. Callan, and J. Broglio. (1993). "TREC-2 routing and ad-hoc retrieval evaluation using the INQUERY system." In *Proceedings of the Third Text REtrieval Conference (TREC-3)*. Gaithersburg, MD: National Institute of Standards and Technology, special publication 500-225, pp. 29-38.

18. W. Hersh, C. Buckley, T. J. Leone and D. Hickam (1994.) OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 192-201. ACM Press.

19. D. A. Hull. (1999.) The TREC-7 Filtering track: Description and analysis. In The Seventh Text Retrieval Conference (TREC-7). National Institute of Standards and Technology, Special Publication 500-242.

20. D A. Hull, and S. E. Robertson. (1999.) The TREC-8 Filtering Track final report. In *Proceeding of the Eighth Text REtrieval Conference (TREC-8),* pp. 35-56. National Institute of Standards and Technology, Special Publication 500-246.

21. W. P. Jones and G. W. Furnas. (1987.) "Pictures of relevance." *Journal of the American Society for Information Science*.

22. Y.H. Kim, S.Y. Hahn, and B.T. Zhang. (2000.) Text filtering by boosting Naive Bayes classifiers. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 168-175. ACM Press.

23. W. Kraaij, R. Pohlmann, and D. Hiemstra. (1999.) "Twenty-One at TREC-8: Using language technology for information retrieval." In *Proceedings of the Eights Text REtrieval Conference (TREC-8).* NIST Special Publication 500-246.

24. J. Lafferty and C. Zhai. (2001.) "Document language models, query models, and risk minimization for information retrieval." In Proceedings of the 24th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR).

25. L. Lee and F. Pereira. (1999.) "Distributional similarity models: Clustering vs. nearest neighbors." *Proceedings of the 37th Annual Meeting of the ACL, 1999.*

26. H. P. Luhn. (1958.) "The automatic creation of literature abstracts." *IBM Journal of Research*, vol. 2, pages 159-165.

27. S. Luo and J. Callan. (2001.) "Named entity detection." Technical Report, Language Technologies Institute, Carnegie Mellon University. February, 2001.

28. R.D. Lyer, D.D. Lewis, R.E. Schapire, Y. Singer, and A. Singhal. (2000.) Boosting for document routing. In *Proceedings of the Ninth International Conference on Information Knowledge Management (CIKM 2000)*, pp. 70-77. ACM Press.

29. D. MacKay. (2001.) "macopt - a nippy wee optimizer." http://wol.ra.phy.cam.ac.uk/mackay/c/macopt.html.

30. http://www.ccr.buffalo.edu/class-notes/hpc2-00/odes/node4.html.

31. R. Manmatha, T. Rath, and F. Feng. (2001.) "Modeling score distributions for combining the outputs of search engines." In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

32. C. Manning and H. Schutze. (1999.) *Foundations of Statistical Natural Language Processing*. MIT Press.

33. A. McCallum, R. Rosenfeld, T. Mitchell, and A. Y. Ng. (1998.) "Improving Text Classification by Shrinkage in a Hierarchy of Classes." *Proceedings of the Eighteenth International Conference on Machine Learning.*

34. D. R. H. Miller, T. Leek, and R. Schwartz. (2001.) "A hidden Markov model information retrieval system." *Proceedings of the 22th Annual International ACM SIGIR Conferenc eon Research and Development in Information Retrieval.*" Pp. 214-221.

35. J. Ponte and W. B. Croft. (1998.) "A language modeling approach to information retrieval." In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 275-281.

36. M. F. Porter. (1980.) An algorithm for suffix stripping. *Program*, 14 (3), pp. 130-137.

37. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. (1992.) *Numerical Recipes in C: The Art of Scientific Computing,* pp. 420-425. Cambridge University Press.

38. S.E. Robertson. (2001.) Threshold setting in adaptive filtering. *Journal of Documentation*.

39. S. E. Robertson, and D. A. Hull. (2001.) Guidelines for the TREC-9 Filtering Track. *Proceeding of Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology, Special Publication.

40. S. Robertson and I. Soberoff. (2002.) "The TREC 2001 Filtering Track report." *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*. National Institute of Standards and Technology, special publication 500-250.

41. S.E. Robertson, and S. Walker. (2001.) Microsoft Cambridge at TREC-9: Filtering track. In *Proceeding of Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology, Special Publication.

42. S. E. Robertson, S. Walker, M. M. Beaulieu, and M. Gatford, A. Payne. (1995.) Okapi at TREC-4. In *Proceeding of Fourth Text REtrieval Conference (TREC-4).* National Institute of Standards and Technology, Special Publication 500-236.

43. J. J. Rocchio. (1971.) "Relevance feedback in information retrieva." In G. Salton, editor, *The SMART retrieval system – Experiments in automatic document processing*, chapter 14. Prentice Hall.

44. R.E. Schapire, Y. Singer, and A. Singhal. (1998.) Boosting and Rocchio applied to text filtering. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 215-223. ACM.

45. C. Zhai, P. Jansen, N. Roma, E. Stoica, and D.A. Evans. (1999.) Optimization in CLARIT adaptive filtering. In *Proceeding of the Eighth Text REtrieval Conference (TREC-8),* pp. 253-258. National Institute of Standards and Technology, Special Publication 500-246.

46. C. Zhai, P. Jansen, E. Stoica, N. Grot, and D. A. Evans. (1999.) "Threshold calibration in CLARIT adaptive filtering. In Proceedings of the Seventh Text REtrieval Conference (TREC-7). NIST Special Publication 500-242.

47. C. Zhai and J. Lafferty. (2001.) "Model-based feedback in the language modeling approach to information retrieval." In *Proceedings of the Tenth International Conference on Information and Knowledge Management.*

48. C. Zhai and J. Lafferty. (2001b.) "A study of smoothing methods for language models applied to ad-hoc information retrieval." *Proceedings of the 24th Annual Int'l ACM SIGIR Conferenc eon Research and Development in Information Retrieval*, pp. 334-342.

49. Y. Zhang and J. Callan. (2001a.) "A generative model for filtering thresholds." *Workshop on Language Modeling and Information Retrieval*, Carnegie Mellon University, May 31-June 1, 2001.

50. Y. Zhang and J. Callan. (2001b.) "Maximum likelihood estimation of filtering thresholds." In *Proceedings of the Twenty Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.

51. Y. Zhang and J. Callan. (2001c.) "YFilter at TREC-9." In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology, special publication 500-249.

52. Y. Zhang and J. Callan. (2002). "The bias problem and language models in adaptive filtering." In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*. National Institute of Standards and Technology, special publication 500-250.

53. Y. Zhang, J. Callan, and T. Minka. (2002.) "Novelty and redundancy detection in adaptive filtering." In *Proceedings of the Twenty Fifth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Tampere, Finland: ACM.

54. Y. Zhang, W. Xu., and J. Callan. (2002.) "Exact maximum likelihood estimation for word mixtures." In *ICML 2002 Workshop on Text Learning (TextML 2002).*